Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

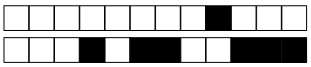
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

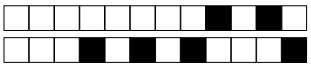
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



+10/5/17+

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

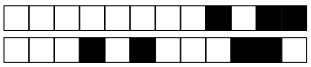
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

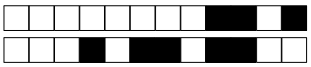
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

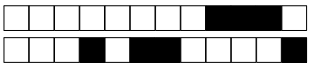
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

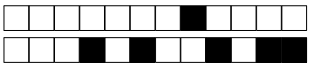
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



+16/5/11+

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

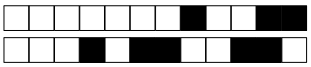
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

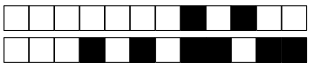
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



+20/5/27+

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

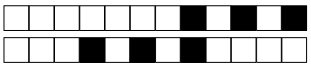
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

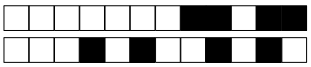
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

- Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
- Les téléphones doivent être rangés.
- Les réponses sont à reporter sur les feuilles de réponse jointes.
- Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
- Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
- Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
- Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

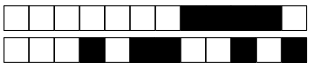
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

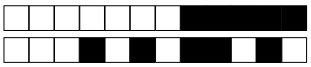
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

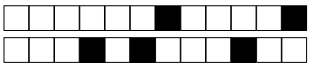
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

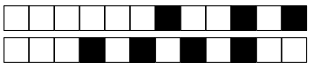
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

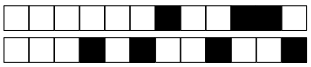
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

+39/5/58+

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0
☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1
☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2
☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3
☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4
☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5
☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6
☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7
☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8
☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

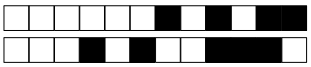
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

- Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
- Les téléphones doivent être rangés.
- Les réponses sont à reporter sur les feuilles de réponse jointes.
- Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
- Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
- Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
- Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

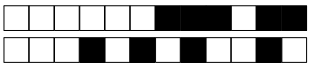
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

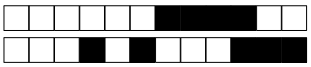
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0
☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1
☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2
☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3
☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4
☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5
☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6
☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7
☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8
☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

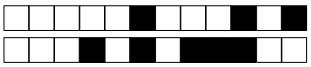
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

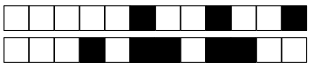
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

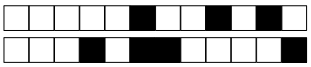
Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```



☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Programmation fonctionnelle

Examen du 11 janvier 2024

L2 math-info / L3 info – Portail Sciences – Université Côte d'Azur

Durée : 2 heures.

— Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
— Les téléphones doivent être rangés.
— Les réponses sont à reporter sur les feuilles de réponse jointes.
— Les réponses doivent être écrites lisiblement. Le correcteur blanc et l'effaceur sont autorisés, ils peuvent être utilisés pour décocher une case cochée par erreur, mais dans ce cas, n'essayez pas de redessiner la case. Vous pouvez déborder du cadre de réponse prévu s'il n'y a pas assez de place, si possible sans déborder sur une autre feuille.
— Les questions sont notées de manière indépendante : une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée. On ne peut pas utiliser la fonction d'une question pour répondre à une question précédente (par exemple, on ne peut pas utiliser la fonction de la question 7 pour répondre à la question 2).
— Il est recommandé de traiter les exercices et les questions dans l'ordre, sans bloquer sur une question. Certaines questions portent une indication de difficulté, forcément subjective.
— Le barème est donné à titre indicatif et pourra être revu pour améliorer la moyenne globale de la promo.

Balise HTLM IMG (4 points)

On modélise une balise IMG avec ses attributs (simplifiés) par le type de données suivant :

```
type widthinfo = Pixels of int | Percents of int
type img = {
   src : string;
   alt : string option;
   width : widthinfo option
}
```

Ce type de données permet de représenter des balises telles que

```
<img src="images/toto.jpg" alt="une photo de toto" width="200px">
<img src="images/toto.jpg" width="80%">
```

Question 1 (2pt) Écrivez une fonction `img_to_string: img -> string` qui renvoie le code HTML correspondant aux informations de la balise IMG.

Question 2 (2pt) Écrivez une fonction `img` avec les labels `src` (obligatoire), `alt` (optionnel), `widthpx` et `widthprct` (aussi optionnels) qui renvoie un enregistrement de type `img` initialisé en fonction des paramètres passés dans les labels. La fonction prend comme dernier argument `()` pour distinguer une application partielle d'une omission de paramètre optionnel. Exemples :

```
img ~src:"images/toto.jpg" ~alt:"une photo de toto" ~widthpx:200 ()
img ~src:"images/toto.jpg" ~widthprct:80 ()
```

La fonction lève une exception à l'aide de `invalid_arg` si les deux labels `widthpx` et `widthprct` sont définis.

Kata list (4 points)

Question 3 (1pt) On considère la fonction `take k l` qui prend en argument une liste `l` et un entier `k` et renvoie la liste des `k` premiers éléments de `l`. Par exemple, `take 2 [1;2;3;4]` renvoie `[1;2]`, ou encore `take 0 ['a';'b';'c']` renvoie `[]`. Si `l` contient moins de `k` éléments, la fonction renvoie `l`. Quel est le type le plus général possible pour `take` ?

Question 4 (1pt) Définissez la fonction `take` de manière récursive. Précisez si votre fonction est récursive terminale ou récursive enveloppée (les deux sont possibles).

Question 5 (1pt) On considère la fonction `zip l1 l2` qui prend en argument deux listes `l1` et `l2` et renvoie la liste des couples formés par les éléments de `l1` et `l2`. Si les deux listes n'ont pas la même longueur, les derniers éléments de la liste la plus longue sont ignorés. Par exemple, `zip [1.;2.] [3;4;5]` renvoie `[(1.,3);(2.,4)]`. Quel est le type le plus général possible pour `zip` ?

Question 6 (1pt) Définissez la fonction `zip` de manière récursive terminale.

Tableaux imbriqués et tenseurs (12 points)

Question 7 (1pt) Qu'affiche le top-level Caml après l'évaluation de l'expression suivante ?

```
# let t = [|1; [|2; 3|]; [|[|4|]|]|];;
```

On considère le type de données suivant :

```
type 'a nested_array =
   Val of 'a |
   Array of 'a nested_array array
```

Ce type de données permet de représenter des tableaux imbriqués, par exemple

```
# let t = Array ([|Val 1;Val 2;Val 3|]);;
val t : int nested_array = Array [|Val 1; Val 2; Val 3|]
# let u = Array ([|Val 'a';Array ([|Val 'b';Val 'c'|]);Val 'd'|]);;
val u : char nested_array = Array [|Val 'a'; Array [|Val 'b'; Val 'c'|]; Val 'd'|]
```

Question 8 (1pt) Définissez par récurrence la fonction `nested_array_length t` qui prend en argument un tableau imbriqué `t` et renvoie le nombre d'éléments de `t`. Par exemple, `nested_array_length u` avec `u` comme ci-dessus renvoie 4.

Question 9 (1pt) Définissez par récurrence la fonction `get: 'a nested_array -> int list -> 'a` qui prend en argument un tableau imbriqué t et une liste d'entiers `is=[i1;...;in]` et renvoie l'élément de `t` correspondant à la suite d'indices `is`, autrement dit "quelque chose comme" `t.(i1).(i2)...(in)` (mais n'oubliez pas qu'il y a les constructeurs `Val` et `Array` à gérer). Si un indice n'est pas valide, ou si la liste `is` n'est pas de la bonne longueur, la fonction lève une exception à l'aide de `invalid_arg`.

Question 10 (1pt) Un vecteur est un tableau imbriqué de dimension 1, c'est-à-dire un tableau imbriqué dont tous les éléments sont des valeurs, et non des tableaux imbriqués. Par exemple, `Array [|Val 1;Val 2;Val 3|]` est un vecteur, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas un. Définissez la fonction `is_vector t` qui renvoie `true` si `t` est un vecteur, et `false` sinon.

Question 11 (1pt) Une matrice est un tableau imbriqué de dimension 2 dont tous les éléments sont des vecteurs de même longueur. Par exemple, `Array [|[|Val 1;Val 2|];Array [|Val 3; Val 4|]|]` est une matrice, mais `Array [|Val 1;Array [|Val 2;Val 3|]|]` n'en est pas une. Définissez la fonction `is_matrix t` qui renvoie `true` si `t` est une matrice, et `false` sinon.

Question 12 (1,5pt) La dimension d'un tableau imbriqué est définie par récurrence :

- `Val x` est un tableau imbriqué de dimension 0
- `Array[|t1;...;tm|]` est de dimension n+1 si tous les ti sont de dimension n.
- si les ti ne sont pas tous de même dimension, le tableau imbriqué n'a pas de dimension.

En particulier, un tableau imbriqué est un vecteur si et seulement s'il est de dimension 1, et toute matrice est un tableau imbriqué de dimension 2 (mais un tableau imbriqué de dimension 2 n'est pas nécessairement une matrice). Définissez la fonction `dimension:'a nested_array -> int option` qui renvoie la dimension de `t` si elle existe, `None` sinon.

Question 13 (1,5pt) La forme d'un tableau imbriqué est une liste d'entiers contenant les tailles du tableau dans chacune de ses dimensions. Par exemple, le vecteur `v = Array [|Val 'a';Val 'b';Val 'c'|]` est de forme `[|3|]`, la matrice `m = Array [|v;v;v;v|]` est de forme `[|4;3|]`. La forme d'un tableau imbriqué n'est pas toujours définie. Par exemple, le tableau imbriqué de dimension 2 `Array[|Array [|Val 1;Val2|]; Array [|Val 3|]|]` n'a pas de forme définie. Une matrice est un tableau imbriqué de dimension 2 dont la forme est définie. On appelle tenseur un tableau imbriqué de dimension quelconque dont la forme est définie. Par exemple, une image RGB de n pixels de hauts sur m pixels de large avec 3 entiers par pixel pourra être représentée par un tenseur de forme `[n;m;3]`. Définissez la fonction `shape:'a nested_array -> int list option` qui renvoie la forme de `t` si elle existe, `None` sinon.

Question 14 (2pt) En utilisant les fonctions `Array.of_list`, `Array.to_list`, `List.map`, et `List.flatten`, définissez la fonction `nested_array_flatten t` qui prend en argument un tableau imbriqué `t` et renvoie le tableau des éléments de `t`. Par exemple, `nested_array_flatten u` avec `u` comme ci-dessus renvoie `[|'a';'b';'c';'d'|]`, et `nested_array_flatten (Val 3.14)` renvoie `[|3.14|]`.

Question 15 (2pt) On considère la fonction `nested_array_copy t1 t2`, de type `'a nested_array -> 'a array -> unit` qui prend en argument un tableau imbriqué `t1` et un tableau unidimensionel Caml `t2` tel que `Array.length t2 = nested_array_length t1`. La fonction ne renvoie rien, mais recopie dans `t2` tous les éléments de `t1`, de sorte que `nested_array_flatten t1 = t2`. On cherche une solution sans recopies inutiles, c'est-à-dire sans créer de nouveaux tableaux ou de nouvelles listes, et sans faire de concaténation de tableaux ou de liste. En particulier on ne peut pas appeler la fonction `nested_array_flatten` de la question précédente. Vous pouvez gagner quelques points si vous expliquez pourquoi appeler `nested_array_flatten` crée des recopies inutiles.

Memo sur les listes

```
utop # List.filter;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
utop # List.filter (fun x -> x > 0) [-1;0;1;2;-3;4];;
- : int list = [1; 2; 4]
utop # List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
utop # List.map (fun x -> x*x) [-1;0;1;2;-3;4];;
- : int list = [1; 0; 1; 4; 9; 16]
utop # List.flatten [[1];[2;3];[];[4]];;
- : int list = [1; 2; 3; 4]
utop # List.flatten;;
- : 'a list list -> 'a list = <fun>
```

Memo sur les tableaux et les boucles for

```
utop # Array.make;;
- : int -> 'a -> 'a array = <fun>
utop # Array.make 10 'a';;
- : char array = [|'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'; 'a'|]
utop # Array.init;;
- : int -> (int -> 'a) -> 'a array = <fun>
utop # let arr = Array.init 10 (fun i -> 9-i);;
val arr : int array = [|9; 8; 7; 6; 5; 4; 3; 2; 1; 0|]
utop # for i=0 to 9 do arr.(i)<-i done;;
- : unit = ()
utop # arr;;
- : int array = [|0; 1; 2; 3; 4; 5; 6; 7; 8; 9|]
utop # let t = [|1;2;3;4|];;
val t : int array = [|1; 2; 3; 4|]
utop # Array.length t;;
- : int = 4
utop # Array.to_list t;;
- : int list = [1; 2; 3; 4]
utop # Array.of_list ['a';'b'];;
- : char array = [|'a'; 'b'|]
utop # Array.for_all;;
- : ('a -> bool) -> 'a array -> bool = <fun>
utop # Array.for_all (fun x -> x > 0) [|1;2;3;4|];;
- : bool = true
utop # Array.for_all (fun x -> x > 0) [|1;2;-3;4|];;
- : bool = false
```

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Écrivez explicitement votre numéro d'étudiant (par précaution) : .....

Les réponses aux questions sont à donner exclusivement sur les feuilles qui suivent : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Réponse à la question 1 : imgtostring ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 2 : toimg ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 3 : typetake ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 4 : take ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 5 : typezip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

Réponse à la question 6 : zip ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

Réponse à la question 7 : toplevel ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 8 : nalength ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 9 : get ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

Réponse à la question 10 : isvector ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 11 : ismatrix ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 12 : dimension ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 13 : shape ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Réponse à la question 14 : naflatten ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

Réponse à la question 15 : nacopy ☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....