Programmation fonctionnelle

Partiel du 7 novembre 2023
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.

Un exercice relativement facile (3 points)

Question 1 (3 points) Définissez une fonction `einstein u v` qui prend en argument deux vitesses et qui calcule leur somme avec la loi d'addition des vitesses en relativité restreinte.

$$\mathsf{einstein}(u, v) = \frac{u + v}{1 + uvc^2}$$

où la vitesse de la lumière $c = 300000\ km.s^{-1}$ est une constante que vous définirez en dehors de votre fonction. Le type de la fonction doit être `float -> float -> float`, et non `float * float -> float`. Autrement dit, on attend une fonction curryfiée dans le style OCaml habituel.

Marabout, bout de ficelle (6 points)

Dans cet exercice, on cherche à écrire une fonction `swinging_trigrammes` qui prend une liste `l` et qui renvoie la liste de ses trigrammes en n'en gardant qu'un sur deux. Par exemple,

— `swinging_trigrammes ["ma"; "ra"; "boud'"; "fi"; "celle"; "de"; "ch'val"]` renvoie `[("ma", "ra", "boud'"); ("boud'", "fi", "celle"); ("celle", "de", "ch'val")]`
— `swinging_trigrammes [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]` renvoie `[(1,2,3); (3,4,5); (5,6,7); (7,8,9)]`

Question 2 (1,5 points) Quel est le type de la fonction `swinging_trigrammes` ?

Question 3 (1,5 points) Définissez la fonction `trigrammes` qui renvoie la liste des trigrammes de la liste passée en argument. Par exemple, `trigrammes ['a'; 'b'; 'c'; 'd'; 'e']` renvoie `[('a','b','c');('b','c','d');('c','d','e')]`.

Question 4 (1,5 points) Définissez la fonction `half_alternated` qui prend en argument une liste `l` et renvoie la liste obtenue en ne gardant qu'un élément sur deux. Par exemple, `half_alternated ['a'; 'b'; 'c'; 'd'; 'e']` renvoie `['a'; 'c'; 'e']`.

Question 5 (1,5 points) En déduire la fonction `swinging_trigrammes` décrite précédemment.

Stock options (3 points)

On considère la fonction `stock_option r v` qui a le comportement suivant : si la référence `r` n'a pas de valeur définie, la fonction lui affecte la valeur `v`, sinon elle lève une exception avec `invalid_argument`. Par exemple

```
# let r = ref None;;
val r : '_weak2 option ref = contents = None
# stock_option r 1;;
- : unit = ()
# !r;;
- : int option = Some 1
# stock_option r 2;;
Exception: Invalid_argument "stock_option".
```

Question 6 (1,5 points) Quel est le type de la fonction `stock_option` ?

Question 7 (1,5 points) Définissez la fonction `stock_option`.

Arg max (2 points)

Question 8 (2 points) Définissez la fonction `argmax: ('a -> 'b) -> 'a list -> 'a option` qui prend en argument une fonction `f` et une liste `l` et qui renvoie l'élément de l qui maximise f.
Par exemple, `argmax (fun x-> x*x) [-1; -3; 2; 0]` renvoie `Some (-3)`. Notez que la fonction renvoie une option ; en effet, si la liste est vide, la fonction renverra `None`.

Petites voitures (3 points)

Question 9 (1 points) Définissez un type `voiture` qui regroupe dans un enregistrement les informations suivantes :

- une marque, qui peut prendre uniquement l'une des trois valeurs `Panhard`, `Matra`, ou `Simca` (vous devez définir un type marque !)
- un modèle, décrit par une chaîne de caractères
- une plaque d'immatriculation, composée d'un préfixe chaîne de caractères, d'un nombre, et d'un suffixe chaîne de caractères, par exemple `("AB", 123, "CD")`.

Question 10 (1 points) Définissez un type `parking` basé sur un `array`. Chaque case du tableau décrit une place de parking, qui peut être libre ou occupée par une voiture.

Question 11 (1 points) Définissez une fonction `garage` qui prend un parking et une voiture et qui met la voiture à une place libre. Si toutes les places sont occupées, la fonction lève l'exception `PlusDePlace`. Vous devez déclarer cette exception.

String of list (3 points)

Question 12 (1 points) Définissez la fonction `concat sep l` qui concatène les chaînes de caractères contenues dans `l` en les séparant avec `sep`.
Par exemple, `concat "**" ["hello"; "world"; "salut"; "terre]` renvoie `"hello**world**salut**terre"`.

Question 13 (1 points) Définissez une fonction `string_of_list f l` qui utilise `f` pour convertir les éléments de `l` en chaîne de caractères et qui renvoie la chaîne ainsi constituée.
Par exemple, `string_of_list string_of_int [1;2;3]` renvoie `"[1;2;3]"`.

Question 14 (1 points) Sans définir de fonction auxiliaire (nommée ou anonyme), autrement dit sans utiliser de `let..in` ou de `fun`, et sans récurrence ou itération (autrement dit sans `let...rec` ni `while` ni `for`), définissez une fonction `string_of_int_list_list` qui renvoie la chaîne de caractères représentant une liste de listes d'entiers.
Par exemple, `string_of_int_list_list [[1;2];[3];[]]` renvoie `"[[1;2];[3];[]]"`.
Vous pouvez utiliser une application partielle de fonction. Vous pouvez aussi, sinon, utiliser `List.map`.

Memo

```
let moyenne x y = (x +. y) /. 2.
val moyenne : float -> float -> float = <fun>

let deux_premiers l = match l with
| [] | [_] -> invalid_arg "deux_premiers"
| hd1 :: hd2 :: _tl -> (hd1, hd2)
val deux_premiers : 'a list -> 'a * 'a = <fun>

let counter = ref 0
val counter : int ref = {contents = 0}

let get_and_incr () =
   counter := !counter + 1;
   !counter
val get_and_incr : unit -> int = <fun>

let rec index x l = match l with
| [] -> raise Not_found
| hd :: _ when x=hd -> 0
| _ :: tl -> 1 + index x tl
val index : 'a -> 'a list -> int = <fun>

let a = [|[1; 2]; [3; 4]|]
val a : int list array = [|[1; 2]; [3; 4]|]

a.(0)
- : int list = [1; 2]

a.(0) <- a.(0) @ a.(1)
- : unit = ()

a
- : int list array = [|[1; 2; 3; 4]; [3; 4]|]

Array.length a
- : int = 2

let l = [1;2;3]
val l : int list = [1; 2; 3]

List.map string_of_int l
- : string list = ["1"; "2"; "3"]

List.map (fun x -> x + 1) l
- : int list = [2; 3; 4]

List.fold_left (+) 0 l
- : int = 6

List.fold_left max (-1) l
- : int = 3

List.fold_left (fun acc x -> acc ^ string_of_int x) "" l
- : string = "123"
```