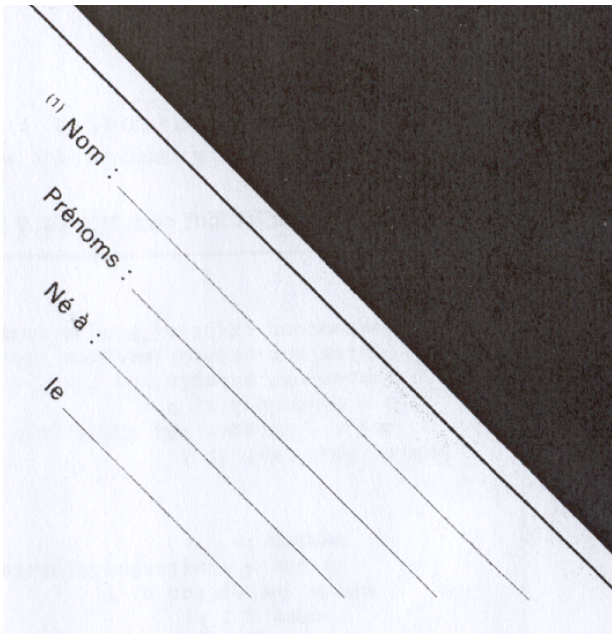
Université de Nice Sophia-Antipolis
L1-MI
Programmation Fonctionnelle I
Examen - Janvier 2014

Durée : 1h30

Seul document autorisé : le formulaire qui vous sera distribué.

Note :



Questions diverses et délectables

1 Soient `L1`, `L2` et `L3` trois listes ayant respectivement `n1`, `n2` et `n3` éléments. Quel est le nombre total d'appels à la fonction **`cons`** lors du calcul de chacune des expressions E et F suivantes [qui retournent le même résultat] ?

`E :` `(append L1 (append L2 L3))` → ☐

`F :` `(append (append L1 L2) L3)` → ☐

Laquelle des deux est la plus efficace : E ou F ? ☐ *[la réponse à cette question ne sera prise en compte que si les deux réponses précédentes sont correctes]*.

2 En lisant le texte d'une fonction récursive Scheme, à quoi voit-on qu'elle est itérative ?

☐

3 Par quoi remplacer `E`, `F` et `G` pour que :

`(apply E (map F (filter G '(les 2 ou 3 bateaux)))) → 24`

E : ☐ F : ☐ G : ☐

4 Programmez une fonction `(tirage)` retournant `#t` ou `#f`. Elle tire au hasard deux entiers dans [1,100000000], et retourne `#t` lorsque ces deux entiers sont *premiers entre eux* [leur PGCD égal à 1].

```
(define (tirage)

)
```

5 Programmez par récurrence une fonction `(élément i L)` retournant l'élément numéro i de la liste L. Le premier élément a pour numéro 0. On supposera que l'élément en question existe. Ex :

`(élément 2 '(un peu trop loin))` → `trop`

```
(define (élément i L)

```

Votre fonction est-elle itérative ? ☐ OUI ☐ NON

Quelle est sa complexité en fonction de i et de n, si la liste L est de longueur n ? O( )

6 Je viens d'entendre à Radio Valrose que la probabilité que deux entiers ≥ 1 tirés au hasard soient *premiers entre eux* est égale à $6/\pi^2$. Pour m'en assurer, je procède N fois à l'expérience suivante : je tire au hasard deux entiers dans [1,100000000] et je compte dans un accumulateur combien de fois j'ai tiré deux entiers *premiers entre eux*, cf question 4. J'en déduis alors la probabilité cherchée, qui n'est autre que le quotient du nombre de succès par le nombre total de tirages. Programmez itérativement la fonction `(proba N)` procédant à cette expérience et retournant la probabilité cherchée. Par exemple `(proba 10000)` → `0.6072`

```
(define (proba N)
  (local [(define (iter
```

7 Nous conviendrons d'appeler **ensemble** une liste d'entiers sans répétition, dont l'ordre n'a pas d'importance. Par exemple `(b a c d)` et `(c b a d)` seront considérés comme des ensembles égaux, tandis que `(a d c d)` n'est pas un ensemble. Programmez la fonction `(inter E F)` prenant deux ensembles E et F, et retournant **l'ensemble intersection** E ∩ F, constitué des éléments qui sont à la fois dans E et dans F. Par exemple :

`(inter '(re sol mi fa) '(si la fa sol))` → `(sol fa)`

```
(define (inter E F)
  (cond
```

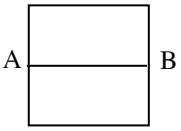
8 ***Changement de variable dans un arbre.*** Programmez la fonction `(subst B1 B2 A)` prenant un arbre binaire d'expression A et retournant une copie de A dans laquelle tout sous-arbre B1 sera remplacé par B2. Par exemple, lorsque le matheux dira « *posons* $x+1=u$ *dans l'expression* $\frac{x+1}{b}-(x+1)$ », le programmeur Scheme dira :

```
(subst '(+ x 1) 'u '(- (/ (+ x 1) b) (+ x 1))) → (- (/ u b) u)
```

```
(define (subst B1 B2 A)                ; remplacer B1 par B2 dans A
  (cond
```

9a ***Animation d'un système de particules***. Dans une scène jaune 300 × 300, une droite noire AB joint les points A(0 ; 150) et B(300 ; 150). Définissez l'image `SCENE` constituée de la scène jaune et de la droite noire.

```
(define SCENE
```



9b Nous allons travailler avec des *billes* représentées par des disques rouges de rayon 10. Définissez l'image `IMG` d'une bille. Toutes les billes partageront la même image.

```
(define IMG
```

9c Une bille est une particule soumise à la gravitation et repérée par sa position `x,y` et sa vitesse `dx,dy`. Elle sera représentée par une *structure* de type `bille` :

```
(define-struct bille (x y dx dy))
```

Au début de l'animation, 30 billes sont placées au hasard sur la ligne AB, avec une vitesse de composante `dx` choisie au hasard dans `{-1,0,1}` et `dy` au hasard dans `{-10,-9,...,0}`. **Le Monde** initial **sera une liste** `INIT` **contenant ces 30 billes aléatoires**. Avec `build-list`, définissez la variable `INIT` :

```
(define INIT
```

9d A chaque top d'horloge, chaque bille va faire un déplacement élémentaire égal à son vecteur vitesse, tout en étant soumise à la gravitation vers le bas [on prendra comme constante gravitationnelle G = 0.5]. Programmez la fonction `(déplacer b)` prenant une bille `b` et retournant la nouvelle bille après le déplacement élémentaire :

```
(define (déplacer b)



```

9e Programmez la fonction `(suivant L)` retournant le monde suivant. On rappelle que le monde est une liste de billes.

```
(define (suivant L)                                    ; Monde --> Monde



```

9f Programmez par récurrence la fonction `dessiner` retournant l'image du monde courant :

```
(define (dessiner L)                                   ; Monde --> Scène



```

9g Programmez la fonction `(final? L)` retournant `#t` lorsque <u>toutes</u> les billes se sont enfoncées dans le sol de la scène.

```
(define (final? L)                                     ; Monde --> Boolean



```

Et pour fêter la nouvelle année, le big-bang est en cadeau :

```
(big-bang INIT
          (on-tick suivant)
          (on-draw dessiner)
          (stop-when final?)
          (name "Bonne Année 2014 !"))
```