Examen de
Programmation fonctionnelle
Durée : 2 h
Tous documents autorisés

Ce sujet comporte 2 pages.

1 Bases

1. Donner les caractéristiques des types suivant ainsi qu'un exemple de valeur pour chacun :
```
type t1 = X;;
type t2 = X | Y of float;;
type 'a t3 = A | B of int * 'a t3;;
type t4 = F of t1 t3 | N of t4 t2;;
```
2. Soient les expression suivantes :
```
#let p = fun n m f x -> n (m f) x;;

#let d = fun g y -> g (g y);;

#let q = p d d;;
```
   (a) Quels sont le type et la valeur de `q` ?

2 Programmation

1. Un nombre parfait est un nombre qui est égal à la somme de ses diviseurs sauf lui-même. Par exemple, 6 est un nombre parfait car 1, 2, et 3 sont ses diviseurs et $6 = 1 + 2 + 3$.
   (a) Écrire une fonction (donner son type) qui vérifie qu'un nombre est parfait.
```
#parfait 6, parfait 7;;
 - : bool * bool = true, false
```
   (b) Écrire une fonction qui énumère en les affichant les nombres parfaits inférieurs à un entier donné.
```
#enumere 1000;;
0
6
28
496
- : unit = ()
```
2. Écrire la fonction `sous_liste` qui vérifie que tous les éléments d'une liste apparaissent dans le même ordre dans une seconde liste.
```
#sous_liste [1;2] [1;3;2;4], sous_liste [1;2;1] [1;1;3;2];;
 - : bool * bool = true, false
```
3. Écrire la fonction `sous_sequence` (donner son type) qui vérifie qu'une liste est la sous-séquence d'une autre. Évaluer la complexité temporelle de votre fonction.
```
#sous_sequence [1;2] [1;3;1;2;4], sous_sequence [1;2] [1;3;1;4];;
 - : bool * bool = true, false
```
4. Soient le type et les fonctions suivantes :
```
#type b = F | N of b * b;;

#let rec f1 = fun n ->
#  if n = 0 then F else N (f1 (n-1), f1 (n-1));;
```

```
#let rec f2 = fun n ->
#  if n = 0 then F else let fn1 = f2 (n-1) in N (fn1, fn1);;
```

(a) Comparer `f1` et `f2` :
- en terme de résultat produit ;
- en terme de temps de calcul ;
- en terme de mémoire utilisée.

(b) Représenter graphiquement sous forme d'arbre le résultat produit par ces fonctions en tenant compte des partages.

3 Mobile

On s'intéresse à la construction de mobiles (ces choses qu'on pend au plafond pour décorer) composés de fils (de masse négligeable), de tiges (de masse également négligeable) et de boules pesantes. À chaque fil (de longueur quelconque) est attachée :
- soit une tige plus ou moins horizontale aux extrémités de laquelle sont attachés deux autres fils ;
- soit une boule.

La figure 1 donne un exemple de mobile.

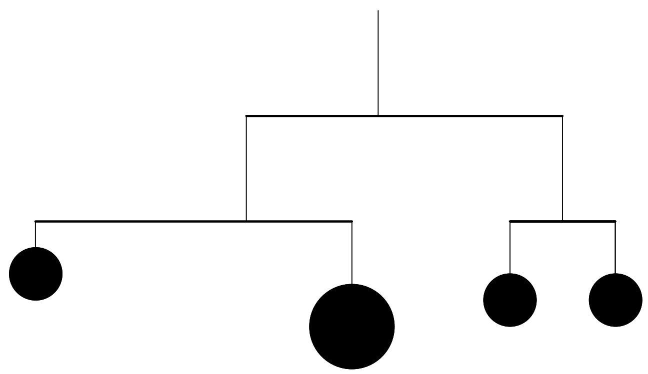

FIG. 1 – Un beau mobile (à peu près) équilibré

Un mobile est dit *équilibré* si toutes les tiges qui le composent sont horizontales (une tige est horizontale si son point d'attache est situé au barycentre des masses qui sont attachées à ses extrémités).

1. Définir le type `mobile` permettant de représenter un mobile (toutes les dimensions et masses seront représentées par des nombres flottants). Représenter le mobile de la figure 1 avec ce type.
2. Définir la fonction qui calcule la hauteur d'un mobile (on néglige la hauteur des boules).
3. Écrire une fonction qui vérifie qu'un mobile est équilibré. Cette fonction prendra en paramètre un flottant ϵ permettant de fixer la précision de l'équilibre : pour l'égalité de deux nombre flottants x et y, on ne testera pas $x = y$ mais $|x - y| < \epsilon$.
4. Écrire une fonction qui teste si deux mobiles sont équivalents, c'est-à-dire ont la même structure et les mêmes dimensions et ne diffèrent que par échange éventuel droite-gauche des deux sous-mobiles des tiges.