Licence STS mention Informatique, semestre 4 — 2016–2017
Introduction à la programmation fonctionnelle — Durée : 2h
2 pages

Examen du 21 juin 2017

Les notes et les transparents de cours sont les seuls documents autorisés.

Question 1 Les fonctions suivantes sont-elles bien typées ? Si oui, donner leur type, sinon préciser pourquoi.

```
let f1 x = (int_of_float x) + 10

let f2 x y = x + y *. x

let f3 x g z = if x then g x else g z

let f4 x y = (x, y)
```

Question 2 Étant donnée la fonction `mystere` suivante, donner (en justifiant votre calcul) la valeur de `v` calculée de la manière suivante :

```
let rec mystere x l =
  match l with
    | [] -> 0
    | y :: s ->
      let v = x + y in
      v + (mystere y s)

let v = mystere 100 [1; 2; 3; 4]
```

1 Programmation : les mots de Dyck

On s'intéresse dans cette partie aux *mots* du *langage de Dyck*. Ce langage est très simple : il s'agit de l'ensemble des mots s'écrivant avec uniquement deux lettres, la parenthèse ouvrante '`(`' et la parenthèse fermante '`)`', et qui sont bien parenthésés, c'est-à-dire qu'à toute parenthèse ouvrante correspond une parenthèse fermante.

Par exemple, le mot `(()())` est bien parenthésé, et appartient donc au langage de Dyck, tandis que le mot `(()(` n'est pas bien parenthésé, et n'appartient donc pas au langage de Dyck.

On représente les mots formés uniquement de parenthèses (bien ou mal parenthésés) à l'aide des types suivants :

```
type parenthese = PO | PF
type mot = parenthese list
```

Les parenthèses ouvrantes et fermantes sont représentées, respectivement, par les constructeurs `PO` et `PF`. Les mots sont simplement représentés par des listes de parenthèses de type `parenthese list`.

Question 3 Donner deux valeurs OCaml de type `mot` pour représenter respectivement les mots `()(())` et `((()(`.

Le but des questions suivantes est de déterminer si un mot appartient au langage de Dyck. Pour cela, on utilise une méthode qui, étant donné un mot `[p1;p2;...;pn]`, calcule la liste d'entiers qui, pour chaque position p_i, donne le nombre de parenthèses ouvrantes non encore fermées à cette position. Par exemple, la liste associée au mot `[PO; PO; PF; PO; PO; PO; PF; PF; PF; PF]` est `[1; 2; 1; 2; 3; 4; 3; 2; 1; 0]`.

Question 4 Écrire une fonction `ouvrante_ou_fermante : int -> parenthese -> int` telle que `ouvrante_ou_fermante cpt p` renvoie `cpt+1` si `p` est une parenthèse ouvrante et `cpt-1` sinon.

Question 5 En utilisant la fonction précédente, écrire une fonction `compte_parentheses : mot -> int list` qui calcule, pour chaque position d'un mot, le nombre de parenthèses ouvrantes non encore fermées. Cette fonction pourra utiliser une fonction auxiliaire récursive `compte` de type `int -> mot -> int list` de sorte que `compte_parenthèses m = compte 0 m`.

On va maintenant chercher à vérifier qu'un mot appartient au langage de Dyck.

Question 6 Écrire une fonction `tous_positifs : int list -> bool` telle que `tous_positifs l` renvoie `true` si et seulement si la liste d'entiers `l` ne contient que des entiers positifs ou nuls. On pourra utiliser un itérateur pour réaliser cette fonction.

Question 7 Compléter le squelette suivant de la fonction récursive `dernier_a_zero : int list -> bool` qui détermine si le dernier entier d'une liste est 0. Il faut remplacer les `...` par du code Ocaml correct.

```
let rec dernier_a_zero l =
  match l with
   | [] -> true
   | [x] -> ...
   | x :: s -> ...
```

Soit `m` un mot et `l` la liste d'entiers renvoyée par `compte_parentheses m`. Le mot `m` appartient au langage de Dyck si et seulement si `l` ne contient que des entiers positifs ou nuls et le dernier élément de `l` est l'entier 0.

Question 8 En utilisant les fonctions précédentes, écrire une fonction `est_un_mot_de_dyck : mot -> bool` qui détermine si un mot appartient au langage de Dyck.

Enfin, on va calculer le *sommet* d'un mot `m`. Ce sommet est défini simplement comme le plus grand entier de la liste renvoyée par `compte_parentheses m`. Par exemple, le sommet du mot `[PO;PO;PF;PO;PO;PO;PF;PF;PF;PF]` donné précédemment est 4.

Question 9 En utilisant la fonction `compte_parentheses`, écrire une fonction `sommet : mot -> int` qui renvoie le sommet d'un mot.