3e année licences d'informatique et 1re année master de linguistique informatique 2007–2008

Examen de Programmation Fonctionnelle

2e session — 17 Juin 2008

Durée : 3h

Documents autorisés : cours, TPs, projets.
Livres interdits — Échange de documents interdit — Téléphones portables interdits

Vous devez répondre aux questions en OCaml, en utilisant uniquement un style fonctionnel (exceptions autorisées). Une fonction écrite en style impératif ne rapportera aucun point.

Le barème ci-dessous est indicatif et est susceptible d'être modifié. Les questions sont indépendantes mais dans le problème il est parfois nécessaire (ou recommandé) d'utiliser les fonctions définies précédemment.

Exercice (8 points)

Répondez (sans justification) aux questions suivantes :

1. Donner le type et la valeur de l'expression suivante :

```
let a = 1000 in
let f = (fun x -> let a = 3 in a + x) in
f 0 + a
```

2. Donner le type et la valeur de l'expression suivante :

```
let g x = 1 in List.map (fun f -> f 4) [(fun x -> x * x); g]
```

3. Quel est le type de la valeur suivante ?

```
let f = function
    | (x, Some w) -> w x
    | _ -> 7
```

4. On définit :

```
type ('a, 'b) blop = Blop of (int, 'a * 'b) blop | Blip of 'b;;
```

Construire une valeur de type `(string, int) blop`.
Quel est le type de `Blop (Blop (Blip (3, ('a', true))))` ?

5. Réécrire la fonction suivante sans utiliser de références et en utilisant une fonction récursive à la place de la boucle. La fonction doit avoir exactement le même type et le même comportement.

```
let somme n =
    let resultat = ref 0 in
    for i = 1 to n do
      print_endline "Entrez un entier :";
```

```
    resultat := !resultat + (read_int ());
  done;
  !resultat
```

6. On tape les commandes suivantes :

```
# let size = 4
size : int = 4
# let t = Array.make size 1;;
val t : int array = [|1; 1; 1; 1|]
# t.(2) <- 6;;
- : unit = ()
# let plie f i t =
    let rec aux first =
      if first = size
      then i
      else f t.(first) (aux (first + 1))
    in
    aux 0;;
```

   Quel est le type de la fonction `plie` ? Quelle est la valeur de `plie (fun x y -> x + y) 0 t` ?

7. Écrire une fonction `taille_suite : 'a -> 'a list -> int` qui, appliquée à une valeur `v` et une liste `l`, renvoie le nombre maximal de valeurs `v` consécutives dans `l`. Par exemple `taille_suite 4 [1; 4; 2; 4; 4; 4; 2]` vaut 3.

8. Expliquer à quoi sert un *garbage collector* (parfois appelé en français *ramasse-miettes* ou *glaneur de cellules*). Donner des exemples de langages avec et sans garbage collector. Quels sont les avantages et inconvénients ? *(une dizaine de lignes)*

Problème - (12 points)

On se propose ici d'écrire un simulateur de robots livreurs de pizzas.

Le monde Le monde est un rectangle composé de cases. On représente les cases par des caractères (voir figure 1). Chaque case est soit vide (`'.'`), soit un mur (`'#'`), soit de l'eau (`'~'`), soit la base (`'@'`). Les robots peuvent se déplacer sur les cases vides. Leur but est de livrer un ou plusieurs clients puis de revenir à la base de départ. Les murs sont infranchissables. Si un robot tombe à l'eau, il se noie et disparaît du jeu. Les bords du monde sont des murs. Les coordonnées du monde sont des paires d'entiers, allant de (1,1) à (*largeur, hauteur*), où (1,1) est le coin sud-ouest du monde.

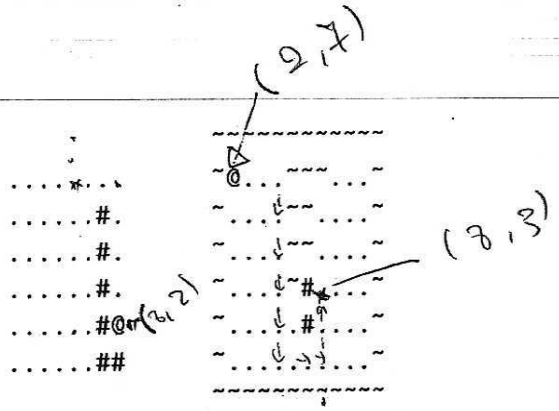
Les robots On contrôle un robot en indiquant la direction du déplacement désiré : Nord, Sud, Est, Ouest. À chaque étape du jeu, chaque robot exécute une commande.

Les clients Un client est repéré par sa position dans le monde.

Vérification de trajectoires

On définit les types suivants :

```
type monde = char array array
type dir = O | E | S | N
type trajectoire = dir list
```



carte 1 carte 2

FIG. 1 – Exemples de cartes

Une trajectoire (suite de déplacement) possible pour livrer une pizza à partir de la base en (8,2) vers la coordonnée (5,8) pour la carte 1 de la figure 1 est : NNNNOOO

1. Sur la carte 2 de la figure 1 donner une trajectoire permettant en partant de la base en (2,7) de livrer une pizza en (8,3).
2. La fonction `trouve_base` prend un monde et retourne les coordonnées de la base ou bien déclenche l'exception `Not_found` s'il n'y a pas de base. Écrire le type de cette fonction, puis la fonction elle-même.
3. La fonction `verifie_trajectoire` prend un monde, une origine, une destination et une trajectoire et indique si la trajectoire est correcte pour relier l'origine à la destination sans rencontrer d'obstacle. Écrire le type de cette fonction, puis la fonction elle-même.

Structure linéaire

Pour les questions suivantes, il y a besoin de construire une structure linéaire homogène (pile, queue ou autre de votre choix). Soit le fichier `st.mli` suivant décrivant l'interface du module `St` :

```
type 'a t
exception Vide
val creer : unit -> 'a t
val ajouter : 'a -> 'a t -> unit
val prendre : 'a t -> 'a
```

Le type `St.t` est abstrait ; la fonction `creer` construit une structure vide ; la fonction `prendre` retourne en l'enlevant (physiquement) un élément de cette structure si elle est non vide ou bien déclenche l'exception `St.Vide` ; enfin la fonction ajouter augmente (physiquement) la structure de l'argument passé.

1. Écrire l'implantation de ce module en indiquant la nature de la structure linéaire que vous décrivez.
2. En fonction de votre implémentation, indiquer ce qu'affiche le programme suivant :

```
let s = St.create();;
St.ajouter 3 s;;
St.ajouter 7 s;;
print_int (St.prendre s);;
print_int (St.prendre s);;
```

Calcul de trajectoires

On cherche à construire une matrice de distances entre les points du monde et une destination. On utilisera la distance Manhantan définie de la manière suivante :

- chaque case a 4 voisines dans les directions O, E, N et S
- chaque voisine d'une case est à une distance de 1 de celle-ci

1. En prenant la base comme destination, on obtient la matrice des distances de tous les points à la base. Donner la matrice des distances pour l'exemple de la carte 1 de la figure 1 en prenant comme destination la base en (8,2).
2. On définit le type `mdist` suivant :

   ```
   type mdist = int option array array
   ```

   On cherche à écrire une fonction prenant un monde, une destination qui construit la matrice des distances par rapport à la destination. On commencera par remplir la case destination (distance 0) puis ses voisines et ainsi de suite. Pour se rappeler des cases à traiter, il est conseillé de les conserver dans une structure linéaire comme une file d'attente. Vous pouvez utiliser le module `St` de la question précédente.

   (a) Écrire une fonction `dist_voisins` qui prend un monde, une matrice des distances, une structure linéaire contenant des coordonnées de cases et les coordonnées d'une case qui possède déjà une distance à la destination qui pour chaque voisin effectue les actions suivantes :
   - si ce n'est pas une case libre : ne rien faire
   - sinon, si la distance de cette case à la destination est déjà connue, ne rien faire
   - sinon lui affecter la distance de la case + 1 et ajouter cette case dans la structure linéaire passée en argument (pour se souvenir qu'il faudra traiter les voisins de cette case plus tard).

   (b) En utilisant la fonction précédente, écrire une fonction `calc_dist` qui prend un monde et une destination puis construit la matrice des distances par rapport à la destination.

Livraison

1. Écrire une fonction livraison qui prend un monde et une destination et retourne une trajectoire de longueur minimale, s'il en existe une, de la base à la destination ou bien déclenche l'exception `Not_found` sinon.
2. On cherche à effectuer plusieurs livraisons en même temps.

   (a) On commence par une première version naïve qui retourne à la base dès qu'une livraison est effectuée puis repart effectuer la suivante. Écrire la fonction `livrer1` qui prend une liste de destinations en repassant par la base entre deux livraisons.

   (b) On propose une deuxième version qui une fois la première livraison effectuée va directement effectuer la seconde sans forcément repasser par la base. Écrire la fonction `livrer2` qui effectue ce travail.

   (c) Proposer sans l'implémenter un algorithme pour une tournée de livraisons plus efficace.