

Examen du 14 mai 2007

Les notes de cours et de TD manuscrites ainsi que les supports de cours distribués cette année sont les seuls documents autorisés.

Veuillez lire attentivement les questions. Veuillez rédiger proprement, clairement et de manière concise et rigoureuse. Le barème est indicatif.

Important. Les types et l'utilisation des fonctions de la bibliothèque standard d'Ocaml mentionnées (ou pouvant être utiles) dans les questions suivantes sont donnés à la fin de ce document.

1 Typage (4 points)

Les fonctions `f` suivantes sont-elles bien typées? Dans l'affirmative, donner leur type, sinon préciser pourquoi.

```
let rec f g x y = if x>y then g x else ()::(g y)
```

```
let rec f x = f x
```

```
let f g x y = if x+1> g y && x = y then g "abc" else g y
```

```
let f x = try
  for i=0 to x do
    if i=2 then raise Exit
  done
with Exit -> ()
```

2 Évaluation (6 points)

Question 2.1 Étant donnée la fonction `f` suivante :

```
let rec mystere x =
  if !x <=1 then () else
  let y = !x in (decr x; mystere x; x := y + !x)
```

Donner le résultat de l'évaluation suivante :

```
let x = ref 5 in (mystere x; !x)
```

Question 2.2 On définit le type `t` ainsi que la fonction `f` de la manière suivante :

```
type 'a t = { mutable a : 'a ; b: 'a }
```

```
let f x = Array.init 2 (fun _ -> {a=x;b=x})
```

Quel est le contenu des variables `t1` et `t2` à la fin de l'évaluation des expressions suivantes ?

```
let t1,t2 = let x = ref 3 in f x , f x
let _ = t2.(0).a <- ref 6
let _ = t1.(0).b :=4
```

Question 2.3 On définit le type `t` ainsi que la fonction `f` de la manière suivante :

```
type t = A of t list list | B of int
```

```
let rec f acc l =
  List.fold_left
    ( fun acc x -> match x with
      B i -> (i*10)::acc
      | A v -> List.fold_left f acc v ) acc l
```

Quel est le résultat de l'appel suivant :

```
f [] [A [[B 2;B 50;A[[B 4]]]] ; B 3]
```

3 Sous-listes (10 points)

Soit l une liste, une *sous-liste* de l est une liste obtenue à partir de l en supprimant zéro, un ou plusieurs éléments (sans changer l'ordre des éléments restants). Par exemple, $[1;5;4;1]$ est une sous-liste de $[3;1;1;5;0;4;1]$.

Question 3.1 Écrire une fonction `est_sous_liste : 'a list -> 'a list -> bool` telle que `est_sous_liste l1 l2` retourne `true` si $l1$ est une sous-liste de $l2$ et `false` sinon.

Le but des trois questions suivantes est de calculer l'ensemble des sous-listes d'une liste. Dans la suite, nous appellerons *ensemble*, une liste sans redondance (*i.e.* ne contenant pas 2 fois le même élément). Afin de distinguer les ensembles des listes dans les types des fonctions suivantes, nous introduisons le type `ensemble` défini de la manière suivante :

```
type 'a ensemble = 'a list
```

Question 3.2 Écrire une fonction `ajout : 'a ensemble -> 'a -> 'a ensemble` qui prend un ensemble `e` et une valeur `v` en argument et qui ajoute `v` à `e`, si `v` n'est pas déjà dans l'ensemble.

Question 3.3 À l'aide de la fonction précédente et d'un itérateur sur les listes, écrire la fonction `union : 'a ensemble -> 'a ensemble -> 'a ensemble` qui prend pour argument deux ensembles `e1` et `e2` et retourne l'union de `e1` et `e2`.

Question 3.4 À l'aide de la fonction précédente et d'un itérateur sur les listes, écrire une fonction `sous_listes : 'a list -> ('a list) ensemble` telle que `sous_liste l` retourne l'ensemble des sous-listes de l .

Nous allons maintenant nous intéresser à la longueur de la plus longue sous-liste commune à deux listes.

Question 3.5 Écrire une fonction `max_liste : int list -> int` qui retourne le plus grand élément d'une liste `l` passée en argument et lève l'exception `Not_found` si `l` est vide.

Question 3.6 À l'aide d'un itérateur sur les listes, écrire la fonction `intersection : 'a ensemble -> 'a ensemble -> 'a ensemble` qui retourne l'intersection de deux ensembles passés en arguments.

Question 3.7 À l'aide des fonctions précédentes, d'un itérateur sur les listes et de la fonction `List.length`, écrire la fonction `longueur_plus_longue_sous_liste : 'a list -> 'a list -> int` tel que `longueur_plus_longue_sous_liste l1 l2` retourne la longueur de la plus longue sous-liste commune à `l1` et `l2`.

Rappels.

`List.map : ('a -> 'b) -> 'a list -> 'b list`

`List.map f [a1; ...; an]` est `[f a1; ...; f an]`

`List.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a`

`List.fold_left f a [b1; ...; bn]` est `f (... (f (f a b1) b2) ...) bn`

`List.filter : ('a -> bool) -> 'a list -> 'a list`

`List.filter p l` retourne tous les éléments de `l` qui satisfont le prédicat `p` .

`List.length : 'a list -> int`

`List.length l` retourne la longueur de la liste `l` .

`Array.init : int -> (int -> 'a) -> 'a array`

`Array.init n f` retourne un nouveau tableau de longueur `n`, où chaque élément d'indice `i` est initialisé avec le résultat de `f i`.