Université Pierre et Marie CURIE

—o—

DEUG MIAS
Types et structures de données
Interrogation écrite

15 janvier 2004

Documents autorisés : tout ce qui tient sur du papier.
Durée : 1 heure 30 minutes.

Remarques :

- *il est parfois nécessaire pour définir une fonction d'avoir à définir d'autres fonctions ou valeurs auxiliaires ;*
- *si besoin est, on peut utiliser une fonction ou valeur demandée à une question précédente sans avoir pour autant répondu cette question.*

Exercice I

Le nombre de combinaisons C_n^p est défini par les trois équations suivantes : pour tous entiers naturels p et n,

$$C_n^0 = 1, \quad C_n^n = 1, \quad C_{n+1}^{p+1} = C_n^{p+1} + C_n^p$$

Question (I.1) Écrivez la définition d'une fonction O'CAML

```
c : int -> int -> int
```

telle que `(c n p)` donne la valeur de $C_{\texttt{n}}^{\texttt{p}}$.

Question (I.2) *Votre* fonction donne-t-elle toujours une valeur ? Pourquoi ?

Exercice II

On considère les *listes d'association*, c'est-à-dire, des listes de type `('a * 'b) list`. Si `kvs` est une liste d'association et que le couple `(k,v)` est un élément de cette liste, on dit que `v` est la *valeur associée* à `k` dans `kvs`. Si `kvs` est une liste d'association et que le couple `(k,v)` est un élément de cette liste, on dit que `(k,v)` est *une association* de `kvs`.

Question (II.1) Donnez la définition d'une fonction O'CAML

```
all_assoc : 'a -> ('a * 'b) list -> 'b list
```

telle que `(all_assoc k kvs)` donne la liste de toutes les valeurs associées à `k` dans `kvs`.

Question (II.2) On dira qu'une liste d'association `kvs` est *stricte* ssi pour tout `k` et pour tout `v` il existe *au plus une* association `(k,v)` dans `kvs`.

Donnez la définition d'une fonction O'CAML

```
add_assoc : 'a -> 'b -> ('a * 'b) list -> ('a * 'b) list
```

telle que si `kvs` est une liste d'association stricte alors `(add_assoc k v kvs)` est une liste d'association stricte qui contient l'association `(k,v)` et toutes les associations `(k',v')` de `kvs` pour `k'`$\neq$ `k`.
Nota : on ne demande pas que l'ordre original soit respecté.

Exercice III Suites triées.

Question (III.1) Donnez la définition d'une fonction O'CAML

```
sorted_list : int list -> bool
```

qui donne la valeur `true` si la liste d'entiers passée en argument est triée en *ordre croissant*.

Question (III.2) Donnez la définition d'une fonction O'CAML

```
sorted_array : int array -> bool
```

qui donne la valeur `true` si le tableau d'entiers passé en argument est trié en *ordre croissant*.

Question (III.3) On définit sur les couples d'entiers les deux ordres suivants :

- $(n, m) \prec (n', m')$ ssi $n + m < n' + m'$ et
- $(n, m) \ll (n', m')$ ssi $n \leq m \leq n' \leq m'$.

Notez qu'une liste peut-être triée selon $\prec$ sans l'être selon $\ll$ mais que toute liste triée selon $\ll$ est aussi triée selon $\prec$.

Donner la définition d'une fonction O'CAML

```
resort : (int * int) list -> (int * int) list
```

qui déclenche une exception `Not_sorted` si la liste passée en argiment n'est pas triée selon $\prec$ ou qui retourne la liste où les arguments ont été réarrangé selon $\ll$.