Université Paris 7 - DEUG-MIAS 29 Mars 2003

Partiel de Logique et Circuits

Documents autorisés : notes de TD et transparents et notes de cours. Durée : 3 heures.
Le barème est donné à titre indicatif et il est susceptible de modifications.

Rédiger chaque exercice sur une feuille indépendante.

Exercice 1 [Total = 5 points] Ecrire les fonctions suivantes en donnant leur type de manière explicite :

1. `nb_occur liste elem` qui calcule le nombre d'occurrences d'un élément `elem` dans une liste `liste`. Par exemple, on a:

```
nb_occur ['a';'b';'a';'f';'a';'g'] 'a' = 3
nb_occur ['d';'e';'h'] 'a' = 0
```

2. `dernier liste` qui retourne le dernier élément de la liste `liste`. Par exemple, on a :

```
dernier ["lundi";"mardi";"mercredi"] = "mercredi"
```

3. `inclus l1 l2` qui teste l'inclusion d'une liste `l1` avec occurrences multiples dans une autre liste `l2`. Attention donc aux occurrences multiples: la liste [1;2;1] est contenue dans la liste [2;1;2;1] mais pas dans la liste [1;2] ni dans la liste [1;2;3], et donc on a :

```
inclus [1;2;1] [2;2;1;1] = true
inclus [1;1;2] [1;2;1]   = true
inclus [1;2;1] [1;2]     = false
inclus [1;2;1] [1;2;3]   = false
```

Pour résoudre cet exercice on suppose l'existence d'une fonction `regroupe liste` qui calcule la liste des couples formés, pour chaque élément distinct de la liste, de cet élément et de son nombre d'occurrences. Par exemple, on a:

```
regroupe [1;2;1;4;1;4] = [(2, 1); (1, 3); (4, 2)]
```

4. `liste_egal liste1 liste2` qui teste l'égalité de deux listes (modulo des permutations des éléments de chaque liste). Par exemple, on a:

```
liste_egal [1;2;4] [4;1;2] = true
liste_egal [1;2]   [2;1;2] = false
liste_egal [5;8]   [4;8]   = false
```

Exercice 2 [Total = 5 points] Considérons les fonctions $\mathtt{m}, \mathtt{d} : \mathbb{Z} \to \mathbb{Z}$ définies par $\mathtt{m}(x) = x + 8$, $\mathtt{d}(x) = x - 11$.

1. Montrez qu'il existe une suite $f_1, ..., f_k$, où chaque f_i est soit `m`, soit `d`, et telle que pour tout $z \in \mathbb{Z}$ on ait $f_k(f_{k-1}(...(f_2(f_1(z)))...)) = z + 1$.

2. Montrez qu'il existe une suite $g_1, ..., g_h$, où chaque g_i est soit `m`, soit `d`, et telle que pour tout $z \in \mathbb{Z}$ on ait $g_h(g_{h-1}(...(g_2(g_1(z)))...)) = z - 1$.

3. En déduire la clôture inductive de $\{0\}$ sous $\{\mathtt{m}, \mathtt{d}\}$.

4. Considèrons les définitions Ocaml:

```
# let m x = x+8;;
val m : int -> int = <fun>
# let d x = x-11;;
val d : int -> int = <fun>
```

Ecrire une fonction `deplacement` de type `int ->int ->((int->int) list)` telle que `deplacement n k` produit comme résultat une liste de fonctions `[h_1;h_2;...;h_k]`, où chaque fonction de la liste est `m` ou `d` et `h_k(...(h_1(n))...) = k`.

(Idée : en utilisant vos réponses aux questions 1 et 2 commencez par définir deux listes

`plus_un: (int->int) list`

`moins_un: (int->int) list`

puis utilisez ces listes `plus_un` et `moins_un` pour écrire la fonction `deplacement`).

Exercice 3 [Total = 5 points]

Pour tout entier strictement positif k, on considère $\mathbb{N}^k = \mathbb{N} \times \mathbb{N} \times \ldots \times \mathbb{N}$ muni de l'ordre produit vu en cours. Une anti-chaîne est un ensemble X telle que, si x et y sont des éléments distincts de X, alors x et y sont incomparables:

$$\forall x, y \in X,\ x \neq y \Rightarrow x \not\leq y \text{ et } y \not\leq x.$$

1. Dans $\mathbb{N}^2$, expliquer pourquoi l'ensemble (1,3), (3,1), (2,4) n'est pas une anti-chaîne.

2. Quelle est le nombre maximal d'éléments d'une anti-chaîne dans $\mathbb{N}$?

3. Dans $\mathbb{N}^3$, donner une anti-chaîne de six éléments contenant $(1, 1, 1)$.

4. Dans $\mathbb{N}^2$, montrer que si X est une anti-chaîne, et que (n, m) est un élément de X, alors X a au plus $n + m + 1$ éléments.

5. Montrer que si X est une anti-chaîne, alors tout sous-ensemble de X est une anti-chaîne.

6. Soit $k > 1$, X une anti-chaîne de $\mathbb{N}^k$ et $x = (x_1, x_2, ..., x_k)$ un élément de X. Pour tout entier i de $[1 \ldots k]$, pour tout entier j de $[0 \ldots x_i - 1]$, on pose l'ensemble:

$$P_{i,j}(x) = \{(y_1, y_2, ..., y_k) \in \mathbb{N}^k \mid y_i = j\}.$$

   (a) Dans $\mathbb{N}^2$, combien y a t-il d'ensembles $P_{i,j}((3, 4))$?
   (b) Dans $\mathbb{N}^k$ et pour un $x \in X$ quelconque, combien y a t-il d'ensembles $P_{i,j}(x)$?
   (c) Exhiber un y incomparable avec $(3, 4)$. Dans quel ensemble $P_{i,j}((3, 4))$ se trouve t-il?

(d) De manière générale, montrer que si $y \in \mathbb{N}^k$, $x \in X$ et y est incomparable avec x, alors y appartient à un certain $P_{i,j}(x)$.

(e) **(Bonus)** Soit X une anti-chaîne et $x \in X$. On suppose que toute anti-chaîne de $\mathbb{N}^{k-1}$ est finie. Montrer que toute anti-chaîne incluse dans un $P_{i,j}(x)$ est finie. En déduire que $X \cap P_{i,j}(x)$ est finie, puis que X est fini.

Exercice 4 [Total = 5 points] Les moines du monastère de Shaolin s'adonnent aux arts martiaux depuis leur plus jeune âge. La hiérarchie est claire : il y a des *maîtres* et des *élèves*. Parmi les maîtres, il y a le *Grand Maître* qui possède le savoir et quelques *maîtres* qui ont un *grade* (un entier entre 1 et 10). Les *élèves* possèdent une ceinture (*Blanche*, *Orange*, *Bleue* ou *Noire*) ainsi qu'un surnom (représenté par une chaîne de caractères).

Les promotions dans le monastère marchent de la façon suivante : un élève promu change de couleur de ceinture (en commençant par *Blanche*, puis *Orange*, puis *Bleue* et enfin *Noire*), un élève de ceinture *Noire* devient un maître de grade 1, et grimpe les grades 1 par 1 jusqu'à 10, et un maître de grade 10 devient *Grand Maître*. Le *Grand Maître* ne peut être promu.

1. Définir un type `couleur` pour représenter les différents types de ceintures.
2. Définir un type abstrait `moine` pour représenter les moines du monastère de Shaolin. Vous donnerez un domaine, les fonctions de construction, les fonctions de test et les fonctions d'accès.
3. Définir une fonction `promotion` qui prend un moine en entrée et renvoie le moine promu. Donner deux codages : le premier abstrait et le deuxième concret.
4. Définir une fonction `plus_fort` prenant deux moines en arguments et renvoyant `true` si le premier moine est plus fort que le second et `false` sinon. Un moine est évidemment plus fort qu'un autre s'il est plus haut dans la hiérarchie.