ISIMA 1ère année

vendredi 23 janvier 2015
Durée : 2 heures

PROGRAMMATION FONCTIONNELLE

Etant donné un ensemble E, on appelle permutation de E toute bijection de E dans E. On considère dans ce problème les permutations d'ensembles finis et ordonnés, c'est-à-dire pour lesquels les éléments peuvent être énoncés dans un ordre donné. De tels ensembles, de type $E = \{ x_1, x_2, \ldots, x_n \}$ seront représentés par une liste `(x1 x2 … xn)`.

Toute permutation s de $E = \{ x_1, x_2, \ldots, x_n \}$ peut être déduite d'une permutation σ de l'ensemble $[\![1, n]\!]$, en appliquant σ aux indices des éléments de E. Par exemple si σ est une permutation de $[\![1, 6]\!]$, telle que $\sigma(1) = 2$, $\sigma(2) = 6$, $\sigma(3) = 1$, $\sigma(4) = 5$, $\sigma(5) = 4$ et $\sigma(6) = 3$, on peut déduire de σ la permutation s de $E = \{ x_1, x_2, \ldots, x_6 \}$ donnée par : $s(x_1) = x_2$, $s(x_2) = x_6$, $s(x_3) = x_1$, $s(x_4) = x_5$, $s(x_5) = x_4$, $s(x_6) = x_3$ (ou, plus généralement : $\forall\, i \in [\![1, n]\!]$, $s(x_i) = x_{\sigma(i)}$).

Toute permutation de l'ensemble $[\![1, n]\!]$ peut être représentée conventionnellement par la liste $(\sigma(1)\ \sigma(2)\ \ldots\ \sigma(n)\ )$. Par exemple, la permutation σ donnée en exemple ci-dessus peut être représentée par la liste `(2 6 1 5 4 3)`. De même, par généralisation, toute permutation s d'un ensemble $E = \{ x_1, x_2, \ldots, x_n \}$ peut être représentée par une liste représentant la permutation σ de $[\![1, n]\!]$ dont elle découle. Attention, cette convention de représentation suppose que les éléments de E sont énoncés dans un ordre bien déterminé. Dans ce problème, cet ordre sera naturellement l'ordre dans lequel les éléments sont écrits dans la liste représentant l'ensemble. Par exemple, si $E = \{ a, b, c \} = \{ c, a, b \}$ = etc. (on a le droit d'énoncer les éléments d'un ensemble dans n'importe quel ordre) est représenté par la liste `(a b c)`, alors la liste `(3 1 2)` représente une permutation s telle que s (`a`) = `c`, s (`b`) = `a` et s (`c`) = `b`.

1°) a) Ecrire une fonction `rang` ayant comme argument un élément `x` et une liste `L` et telle que l'évaluation de l'expression `(rang x L)` retourne 0 si `x` n'appartient pas à `L` et sinon retourne le rang de `x` dans `L`.

b) Ecrire une fonction `nieme` ayant comme argument un entier positif `n` et une liste `L` et telle que l'évaluation de l'expression `(nieme n L)` retourne le `n`ème élément de `L` si `n` est inférieur ou égal à la longueur de `L` et retourne `#f` sinon.

c) Ecrire une fonction <u>récursive terminale</u> `list1n` ayant comme argument un entier positif `n` et telle que l'évaluation de l'expression `(list1n n)` retourne la liste des entiers de 1 à `n`, dans l'ordre croissant.

2°) a) Ecrire une fonction `sigma` ayant comme argument une liste `LP` représentant une permutation et telle que l'évaluation de l'expression `(sigma LP)` retourne la permutation de l'ensemble $[\![1, n]\!]$ représentée par `LP`.

Par exemple l'évaluation de `(sigma '(2 6 1 5 4 3))` doit retourner la permutation σ donnée en exemple ci-dessus, c'est-à-dire la fonction qui à 1 associe 2, à 2 associe 6, etc. Donc, l'évaluation de l'expression `((sigma '(2 6 1 5 4 3)) 2)` doit par exemple retourner `6`.

b) Ecrire une fonction `permut` ayant comme arguments une liste `E` représentant un ensemble E, et une liste `LP` représentant une permutation de E, et telle que l'évaluation de l'expression `(permut E LP)` retourne la permutation de l'ensemble E représentée par `LP`.

Par exemple l'évaluation de `(permut '(a b c) '(3 1 2))` doit retourner la permutation s telle que s (`a`) = `c`, s (`b`) = `a` et s (`c`) = `b`.

c) Ecrire une fonction `liste` ayant comme arguments une liste `E` représentant un ensemble E, et une permutation `s` de E, et telle que l'évaluation de l'expression `(liste E s)` retourne la liste représentant `s`.
Par exemple l'évaluation de l'expression

```
(let ((s (permut '(a b c) '(3 1 2))))
   (liste '(a b c) s))
```

→ (c a b)

doit retourner `(3 1 2)`.

3°) Ecrire une fonction `image` ayant comme arguments une liste `E` représentant un ensemble E, et une liste `LP` représentant une permutation de E, et telle que l'évaluation de l'expression `(image E LP)` retourne l'image de l'ensemble E par la permutation représentée par `LP`.
Par exemple l'évaluation de `(image '(a b c) '(3 1 2))` doit retourner `(c a b)`.

4°) a) Ecrire une fonction `inverseListe` ayant comme argument une liste `LP` représentant une permutation σ et telle que l'évaluation de l'expression `(inverseListe LP)` retourne une liste représentant la permutation inverse σ^{-1}, définie par : $\sigma(x) = y \Leftrightarrow x = \sigma^{-1}(y)$.
Par exemple l'évaluation de `(inverseListe '(2 6 1 5 4 3))` doit retourner `(3 1 6 5 4 2)`.
<u>Remarque</u> : $\forall i \in$ `LP`, $\sigma^{-1}(i)$ est le rang de i dans `LP`.

b) Ecrire une fonction `inverse` ayant comme arguments une liste `E` représentant un ensemble E et une permutation `s` définie sur E, et telle que l'évaluation de l'expression `(inverse E s)` retourne la permutation inverse s^{-1} de E.

5°) Soit $E = \{ x_1, x_2, \ldots, x_n \}$ un ensemble et $k \in [\![2, n]\!]$. On appelle k-cycle de E toute permutation c de E telle qu'il existe $x_{i_1}, \ldots, x_{i_k} \in E$, deux à deux distincts, tels que $c(x_{i_1}) = x_{i_2}$, $c(x_{i_2}) = x_{i_3}$, … , $c(x_{i_{k-1}}) = x_{i_k}$, $c(x_{i_k}) = x_{i_1}$ et $\forall x \in E - \{ x_{i_1}, \ldots, x_{i_k} \}$, $c(x) = x$. Un théorème mathématique démontre que toute permutation peut être décomposée en un ou plusieurs cycle(s).
Par exemple, la permutation de $[\![1, 6]\!]$ représentée par la liste `(2 6 1 5 4 3)` se décompose en deux cycles c_1 et c_2 tels que :

$$c_1(2) = 6,\ c_1(6) = 3,\ c_1(3) = 1 \text{ et } c_1(1) = 2 \quad ; \quad c_2(4) = 5 \text{ et } c_2(5) = 4$$

Chaque cycle c_1 et c_2 pouvant être respectivement représentés par la liste `(2 6 3 1)` et `(5 4)`, la décomposition de la permutation donnée en exemple peut être représentée par la liste de listes `((2 6 3 1) (5 4))`.

a) Ecrire une fonction `cycleIteres` ayant comme arguments une fonction `f` d'une seule variable et un élément `x`, et telle que l'évaluation de l'expression `(cycleIteres f x)` retourne une liste $(x_1\ x_2\ \ldots\ x_n)$ telle que : $x_1 = $ `x`, $\forall i \in [\![1, n-1]\!]$, `f` $(x_i) = x_{i+1}$, et `f` $(x_n) = x_1$.
Par exemple si `f` est la permutation de $[\![1, 6]\!]$ représentée par la liste `(2 6 1 5 4 3)`, alors l'évaluation de l'expression `(cycleIteres f 2)` doit retourner `(2 6 3 1)`, et l'évaluation de l'expression `(cycleIteres f 5)` doit retourner `(5 4)`.

b) Ecrire une fonction `listeCycles` ayant comme argument une liste `LP` représentant une permutation σ et telle que l'évaluation de l'expression `(listeCycles LP)` retourne une liste de listes représentant la décomposition en cycles de σ. Par exemple, l'évaluation de l'expression `(listeCycles '(2 6 1 5 4 3))` doit retourner `((2 6 3 1) (5 4))` ou toute autre liste de listes équivalente.