Programmation fonctionnelle

Répétition 7

23 avril 2015

Correction exercices proposés

Exercice 1.

Un arbre n-aire est un arbre dont chaque nœud admet un nombre quelconque de fils ; seules les feuilles sont étiquetées, l'étiquette étant un nombre naturel. La fonction f prend comme argument un arbre n-aire a et renvoie l'arbre n-aire $f(a)$ obtenu en remplaçant chaque feuille de a étiquetée $n > 0$ par un nœud interne dont les n fils sont des feuilles étiquetées $n - 1$. On demande de programmer la fonction f.

Exercice 2.

Nous convenons de représenter un *graphe orienté* comportant n nœuds par une liste de n listes ; la liste correspondant au nœud x a pour premier élément x et pour éléments suivants les successeurs (immédiats) de x.

Écrire le prédicat `(path? n1 n2 g)` prenant deux nœuds $n1$ et $n2$ et la représentation g d'un *graphe orienté* en arguments et retournant `#t` s'il existe un chemin de $n1$ vers $n2$ dans g et `#f` sinon.

Exercice 3.

Écrire une fonction `longest-inc` retournant la plus longue sous-liste croissante de la liste d'entiers donnée en argument.

Exercices sur les listes de listes

Exercice 4.

Écrire une fonction `lpref` prenant comme argument une liste `u` et retournant la liste des préfixes de `u`. (La liste vide et la liste `u` elle-même sont des préfixes de `u`.)

Exercice 5.
Une *tricoupure* d'une liste ℓ est une liste de trois listes non vides dont la concaténation (dans l'ordre) vaut ℓ. Écrire une fonction `tricoup` qui à toute liste ℓ associe la liste des tricoupures de ℓ.

Par exemple, si ℓ est `(a b)` la liste des tricoupures de ℓ est la liste vide ; si ℓ est `(a b c d)` la liste des tricoupures de ℓ comporte, dans un ordre quelconque, les trois listes `((a b) (c) (d))`, `((a) (b c) (d))` et `((a) (b) (c d))`.

Variante

Une *tricoupure* d'une liste ℓ est une liste de trois listes dont la concaténation (dans l'ordre) vaut ℓ. Écrire une fonction `tricoup-lv` qui à toute liste ℓ associe la liste des tricoupures de ℓ.

Par exemple, si ℓ est `(a)` la liste des tricoupures de ℓ comporte, dans un ordre quelconque, les trois listes `((a) () ())`, `(() (a) ())` et `(() () (a))`.

Exercice 6.
Écrire une fonction qui renvoie la liste des sous-ensembles d'un ensemble donné.

`(lset '(a b c))` $\Rightarrow$ `(() (a) (a b) (a b c) (a c) (b) (b c) (c))`

Exercice 7.
Écrire une fonction qui renvoie la liste des permutations d'une liste donnée.

Exercices proposés

Exercice 8.
Écrire une fonction `nbsum` qui prend comme argument un nombre n et qui renvoie le nombre de façons d'écrire une somme égale à n (on comptera une seule fois les commutations).

Exercice 9.
Écrire une fonction qui prend en argument une liste et renvoie cette liste dont on a supprimé 3 éléments sur 5. On commence par garder les deux premiers éléments, puis on supprime les trois suivants, puis on garde les deux suivants, ...

Exercice 10.

Générer à partir d'une liste, tous les arbres binaires complets dont la lecture des feuilles de gauche à droite donne la liste.

Exercice 11.

Écrire la fct `lagrange`, prenant comme argument un entier naturel n et renvoyant la liste de tous les quadruplets d'entiers naturels (a, b, c, d) tels que $a^2 + b^2 + c^2 + d^2 = n$.

```
(lagrange 13) ==>
((0 0 2 3) (0 0 3 2) (0 2 0 3) (0 2 3 0) (0 3 0 2) (0 3 2 0)
 (1 2 2 2) (2 0 0 3) (2 0 3 0) (2 1 2 2) (2 2 1 2) (2 2 2 1)
 (2 3 0 0) (3 0 0 2) (3 0 2 0) (3 2 0 0))

(lagrange 18) ==>
((0 0 3 3) (0 1 1 4) (0 1 4 1) (0 3 0 3) (0 3 3 0) (0 4 1 1)
 (1 0 1 4) (1 0 4 1) (1 1 0 4) (1 1 4 0) (1 2 2 3) (1 2 3 2)
 (1 3 2 2) (1 4 0 1) (1 4 1 0) (2 1 2 3) (2 1 3 2) (2 2 1 3)
 (2 2 3 1) (2 3 1 2) (2 3 2 1) (3 0 0 3) (3 0 3 0) (3 1 2 2)
 (3 2 1 2) (3 2 2 1) (3 3 0 0) (4 0 1 1) (4 1 0 1) (4 1 1 0))
```