INFO0054 - Programmation fonctionnelle

Répétition 3: Récursion sur les nombres et sur les listes

Jean-Michel Begon

27 février 2018

Récursion sur les listes

Exercice 1.
Ecrire une fonction `double` qui va doubler les nombres de la liste qu'elle prend en entrée.

Exercice 2.
Écrire une fonction `remove-all` à deux arguments `l` et `n` dont les valeurs sont respectivement une liste et un nombre entier, qui retourne la liste privée de toutes les occurrences de `n`.

`(remove-all '(2 7 1 7 3 1) 1)` ⇒ `(2 7 7 3)`

Exercice 3.
Définir la fonction `filter_` à deux arguments, un prédicat unaire `p?` et une liste d'éléments appartenant à son domaine `ls`, et renvoyant la liste des éléments de `ls` pour lesquels l'application du prédicat est `#t`.
Remarque : la fonction `filter` est prédéfinie, nous la redéfinissons sous un autre nom à titre d'exercice.

Exercice 4.
Redéfinir les fonctions `big` et `remove-all` à l'aide de la fonction `filter`.

Spécification

Exercice 5.
Soient les fonctions

```
(define ml
  (lambda (n x)
    (ml-aux n x '())))

(define ml-aux
    (lambda (n x acc)
        (if (zero? n) acc
            (ml-aux (- n 1) x (cons x acc)))))
```

Corriger la spécification suivante :

> Si `n` est un naturel, `x` est un naturel quelconque et `acc` est la liste vide, alors `(ml-aux n x acc)` renvoie une liste de taille `n` dont chaque élément est `x`.

se rapportant à la fonction `ml-aux`.

Exercice 6.
Soient les fonctions suivantes :

```
(define xy
  (lambda (x y)
    (xyz x y 0)))

(define xyz
  (lambda (x y z)
    (cond ((null? y) z)
          ((= x (car y)) (xyz x (cdr y) (+ 1 z)))
          (else (xyz x (cdr y) z)))))
```

Ecrire une spécification pour la fonction `xyz`.

Récursion sur les nombres

Exercice 7.
Les nombres de Gribomont sont définis comme suit :

si $n < 3$, $f(n) = n$
si $n \geq 3$, $f(n) = f(n-1) + f(n-2) + f(n-3)$.

Écrire une fonction qui permet de calculer les nombres de Gribomont.

Exercice 8.
Calculer les coefficients binomiaux à l'aide de la relation $(0 \leq k \leq n)$

$$C_n^k = \begin{cases} 1, & \text{si } k = 0 \text{ ou } k = n \\ C_{n-1}^{k-1} + C_{n-1}^k, & \text{sinon} \end{cases}$$

Spécification

Exercice 9.

Soit la fonction f définie sur $\mathbb{N}$ par :

$$f(n) = \left(\sum_{i=0}^{n-1} ([2 + f(i)] \times [3 + f(n - i - 1)]) \right) \mod (2n + 3)$$

implémentée par le programme suivant :

```
(define f (lambda (n) (fx n 0 '())))

(define fx
  (lambda (k i u)
    (let ((next
           (modulo (apply + (map (lambda (x y) (* (+ 2 x) (+ 3 y)))
                                 u
                                 (reverse u)))
                   (+ i i 3))))
      (if (zero? k) next
          (fx (- k 1) (+ i 1) (cons next u))))))
```

Spécifier la fonction `fx`.