INFO0054 - Programmation fonctionnelle

Répétition 2 : Récursion sur les nombres et sur les listes

Jean-Michel Begon

23 Février 2017

Exemples

Exercice 1.
Écrire une fonction `sumOfIntegers` prenant en argument un naturel n et calculant la somme des naturels inférieurs ou égaux à n.

Exercice 2.
Définir la fonction `sumList` qui calcule la somme des éléments d'une liste de nombres.

Exercices libres

Exercice 3.
Définir la fonction `mod` qui renvoie le modulo de deux nombres.

Remarque : la fonction *modulo* est prédéfinie, nous la redéfinissons sous un autre nom à titre d'exercice.

Exercice 4.
Définir une fonction `pgcd` qui renvoie le plus grand commun diviseur de deux nombres.

Exercice 5.
Écrire une fonction prenant en argument deux nombres x et n et renvoyant x^n.

Remarque : la fonction *expt* est prédéfinie, nous la redéfinissons sous un autre nom à titre d'exercice.

Exercice 6.
Définir la fonction `min` qui renvoie le minimum de la liste non vide de nombres donnée en argument.

Exercice 7.
Définir la fonction `length-list` qui renvoie la longueur de la liste passée en argument.

Remarque : la fonction *length* est prédéfinie, nous la redéfinissons sous un autre nom à titre d'exercice.

Exercice 8.
Définir la fonction `reverse-list` qui prend en argument une liste *ls* et renvoie une liste contenant les éléments de *ls* dans l'ordre inverse.

Remarque : la fonction *reverse* est prédéfinie, nous la redéfinissons sous un autre nom à titre d'exercice.

Réflexions

Exercice 9.
Spécifier la fonction suivante :

```
(define xxx
    (lambda (u)
        (if (null? u) 0
            (if (> (car u) 0)
                (+ (car u) (xxx (cdr u)))
                (xxx (cdr u))))))
```

Exercice 10.
Quel serait le cas de base d'une fonction qui calcule le produit des éléments d'une liste de nombres ?

Accumulateurs

Exercice 11.
Ecrire et spécifier une version *tail-recursive* des fonctions :

- `minimum`
- `length`
- `reverse-list`