DEUG 1^{ere} année
MIAS 1 et MASS 1 – Programmation fonctionnelle

TP n° 5 : Fonctions calculant plusieurs résultats – Récursivité profonde

Dans les exercices qui suivent vous détaillerez les spécifications en donnant explicitement la variable d'induction, la (les) condition(s) d'arrêt les équations de récurrences permettant de résoudre les problèmes posés.

Exercice 1 : Étant donné une liste quelconque, on désire écrire une fonction **symbnomb** qui renvoie le nombre de symboles et le nombre de nombres que contient cette liste au premier niveau. Le résultat de **symbnomb** sera une liste dont le premier élément est le nombre de symboles et le second le nombres de nombres.
Spécifier et écrire la fonction **symbnomb** de sorte qu'elle n'examine chaque élément de la liste qu'une seule fois.
`(symbnomb '(a 2 3 (b 4 5) "toto") )` renvoie `(1 2)`
`(symbnomb '("a" 2 3 (4 5)) )` renvoie `(0 2)`

Exercice 2 : Spécifier et écrire une fonction **separe-pair-impair** qui prend une liste plate d'entiers et renvoie une liste formée de la liste des nombres pairs et de la liste des nombres impairs.
`(separe-pair-impair '(1 2 3 4 5 6 7))` renvoie `((2 4 6)(1 3 5 7))`

Exercice 3 : Spécifier et écrire une fonction **truc** qui prend une liste plate d'entiers et renvoie une liste formée du nombre d'entiers strictment positifs et du nombre de multiples de 3.
`(truc '(1 -4 0 6 0 3 -9 7 12))` renvoie `(5 6)`

Exercice 4 : Spécifier et écrire une fonction **nbatomes** qui compte le nombre d'atomes d'une liste quelconque. On pourra utiliser le prédicat prédéfini **atom?** qui rend vrai si la valeur de son argument est un atome. **Attention !** la liste vide est un atome...
`(nbatomes '( a ( b c (2 3)) 4 () ((f))))` renvoie 8

Exercice 5 : Spécifier et écrire une fonction **aplatir** qui prend une liste quelconque l et renvoie une liste à plat formée des atomes de l
`(aplatir '( a ( b c (2 3)) 4 (((f)))))` renvoie `(a b c 2 3 4 f)`

Exercice 6 : Spécifier et écrire une fonction **sauf** qui supprime toutes les occurences d'un objet donné à tous les niveaux d'une liste.
`(sauf 1 '(2 3 1 (4 1) 2 1 (1)))` renvoie `(2 3 (4) 2 ())`

Exercice 7 : Spécifier et écrire une fonction **enleve** qui prend en arguments deux listes $l1$ (qui est une liste plate) et $l2$ et renvoie la liste $l2$ privée de toutes les occurences de tous les éléments de $l1$ à tous les niveaux.
`(enleve '(1 2 3) '(1 (2 3) 4 3 (2 4) 1))` renvoie `(() 4 (4))`

Exercice 8 : Spécifier et écrire une fonction **nblistes** qui renvoie le nombre de listes contenues dans une liste.
`(nblistes '( a b (c (d e) () f) ((g))))` renvoie 5

Exercice 9 : Spécifier et écrire une fonction **profondeur** qui renvoie la profondeur d'une liste. (La profondeur d'une liste plate est 1)
`(profondeur '(a (b c) d))` renvoie 2
`(profondeur '(a (b c) (d (e () f))))` renvoie 4

Exercice 10 : On considère l'ensemble des expresssions arithmétiques infixées défini par :

- tout symbole de variable est une expression arithmétique
- tout nombre est une expression arithmétique
- toute expression de la forme `(op e1 e2)` où `op` est un des opérateur binaires `+, -, /, *` et où `e1` et `e2` sont des expressions arithmétiques est une expressions arithmétique

Spécifier et écrire une fonction **infixprefix** qui prend en argument une expression arithmétique infixée et renvoie la forme préfixée de cette expression.
`(infixprefix '(a + (b * 12)))` renvoie `(+ a (* b 12))`

Exercice 11 : Écrire une deuxieme version **separe-pair-impair2** de la fonction de l'exercice 2 mais qui opère sur une liste quelconque.
`(separe-pair-impair2 '(1 a c -2 3.7 (3 5 (n 9)) 6 7))` renvoie `((-2 6)(1 3 5 9 7))`