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

TD n° 4

But : Écriture de fonctions récursives.

Idée générale : notons n la taille d'un problème P_n à résoudre (n peut être la taille d'une liste sur laquelle porte le problème par exemple). Le principe de résolution par récurrence nous dit que :

si on sait résoudre le problème pour de petites tailles

et si on sait calculer la solution du problème P_{n+1} en fonction des solutions P_k avec $k \leq n$

alors on sait résoudre le problème quelque soit la taille.

Il faut impérativement respecter les deux règles suivantes lorsqu'on écrit une fonction récursive :

1. Une fonction récursive doit être définie à l'aide d'une expression conditionnelle dont au moins l'un des cas mène a une expression évaluable sans appel récursif. C'est la **condition d'arrêt**.
2. Quelque soit la valeur du (ou des) argument(s) de la fonction, il faut que la condition d'arrêt soit atteinte en un**nombre fini d'appels récursifs**.

Notation : pour donner les spécifications de fonctions manipulant des listes, on pourra utiliser les notations : *listevide* pour désigner la liste vide, *l=(x . l')* pour exprimer que `(car l) = x` et `(cdr l) = l'`, et *(x . l)* pour exprimer que l'on effectue l'opération `(cons x l)`.

Exercice 1 : Définir en scheme une fonction permettant de calculer n'importe quel terme de la suite U définie de manière récurrente par :

$$\begin{cases} U_0 = 1 \\ U_n = 3.U_{n-1} + 2, \quad si \quad i > 0 \end{cases}$$

Exercice 2 : Donner la spécification et définir en scheme une fonction **ajoutetous** qui prend en paramètres un nombre n et une liste plate de nombres et ajoute n à chaque nombre de la liste.

`(ajoutetous 10 '( 1 2 3 4 5 6))` $\longrightarrow$ `(11 12 13 14 15 16)`

Exercice 3 : Donner la spécification et définir en scheme une fonction **tsd** qui prend une liste non vide et renvoie une liste équivalente privée de son dernier élément.

`(tsd '(a d f g h))` $\longrightarrow$ `(a d f g)`

Exercice 4 : Donner la spécification et définir en scheme une fonction **Nbocc** qui renvoie le nombre d'occurences d'un objet dans une liste plate.

`(Nbocc 1 '(1 1 c 1 4 h))` $\longrightarrow$ `3`