TP6 – Algorithmes sur les Listes / Les procédures

Dans ce TP, nous allons nous familiariser avec la **récursion terminale**, les **accumulateurs** qui font références aux chapitre 1.4, 1.6, 3.4.2, 3.4.3.

Nous allons aussi commencer à aborder la notion de **procédure** que l'on peut trouver chapitre 2.3.4 et 2.4 du livre.

1. Algorithmes sur les Listes :

La récursion terminale

La machine abstraite exécute une procédure récursive-terminale avec une taille de pile constante. Cette propriété s'appelle l'optimisation du dernier appel ou l'optimisation terminale (last call optimization). Pour cela, on fait un seul appel récursif qui est le dernier appel dans le corps de la procédure.

```
% Sans récursion terminale
fun {Sum1 N}
if N==0 then 0 else N+{Sum1 N-1} end
end

% Avec récursion terminale
fun {Sum2 N S}
if N==0 then S else {Sum2 N-1 N+S} end
end
```

Question 1 : Réaliser une fonction qui compte récursivement le nombre d'éléments dans une liste !

Question 2 : La fonction créée suit-elle la récursion terminale ? Si ce n'est pas le cas modifié la pour suivre ce principe !

Question 3 : Réaliser une fonction qui fait la somme des éléments d'une liste de manière récursive !

Question 4 : La fonction créée suit-elle la récursion terminale ? Si ce n'est pas le cas modifié la pour suivre ce principe !

Question 5 : Pour aller plus loin, réaliser une fonction qui fait la moyenne des éléments d'une liste en respectant le principe de récursion terminale (indice : il va falloir utiliser l'instruction division entière **div**).

Les accumulateurs

Question 6 : Réaliser une fonction qui fait la moyenne des termes d'une liste de manières récursive.

Question 7 : Réaliser une fonction qui fait la moyenne des termes d'une liste de manières récursive en utilisant les accumulateurs.

Question 8 : Les deux premières questions étant faite, nous allons pouvoir vraiment utiliser les accumulateurs. Je vous demande d'écrire une fonction `Fibonacci` telle que `{Fibonacci N}` renvoie la liste des `N` premiers nombres de Fibonacci. On suppose que `N`>1 et que les deux premiers nombres de la suite sont 1 et 1.

```
{Browse {Fibonacci 10}}   % affiche [1 1 2 3 5 8 13 21 34 55]
```

Pour rappel, chaque nombre de la suite est la somme des deux nombres précédents.

Les procédures :

Question 9 : Écrire une procédure P de valeur égale à celle du programme suivant:

```
declare B=2 fun {Mult A} A*B end
declare B=5
{Browse {Mult 11}}
```

Question 10 : Pour le 24 décembre, la police organise des contrôles d'alcoolémies. La technique est simple : pour tout conducteur ayant plus de 0.25, le contrevenant reçoit une amende sur laquelle est inscrit le lieu de l'infraction. Écrivez le programme qui permet de réaliser le contrôle et en cas d'amende d'afficher le lieu de l'amende.

Instruction : le programme ne peut avoir recourt à plusieurs fonctions différentes !