TP : Les listes en pratique

1) Générateur aléatoire

Commencez par récupérer le fichier disponible à l'adresse suivante (il vous suffit de cliquer sur la séance correspondant à ce TP puis sur *Fonctions pré-implémentées*) :

http://andre.lovichi.free.fr/teaching/ea/2012-2013

Ce fichier contient une fonction `creer_liste_aleatoire` de type `int -> int list`, telle que `creer_liste_aleatoire n` renvoie une liste composée de n éléments choisis au hasard entre 0 et 100.

Evaluez les lignes de code correspondantes afin de pouvoir utiliser cette fonction, et poursuivez votre TP à la suite de cette première fonction.

2) Créations récursives

2.a) De 1 à n

Pour commencer, écrivez une fonction `ordre_croissant` :

- prenant en argument un entier `n` (supposé strictement positif)
- renvoyant une liste
- telle que `ordre_croissant n` renvoie la liste `[ 1 ; 2 ; … ; n ]`

2.b) De 1 à n^2

Imaginez ensuite une fonction `carres_ordre_croissant` :

- prenant en argument un entier `n` (supposé strictement positif)
- renvoyant une liste
- telle que `carres_ordre_croissant n` renvoie la liste `[ 1 ; 4 ; … ; n² ]`

2.c) De n à 1

Ecrivez enfin une fonction `ordre_decroissant` :

- prenant en argument un entier `n` (supposé strictement positif)
- renvoyant une liste
- telle que `ordre_decroissant n` renvoie la liste `[ n ; … ; 2 ; 1 ]`

3) Recherche, insertion et suppression

3.a) Recherche dans une liste non triée

Pour commencer, écrivez une fonction `appartient` :

- prenant en argument une liste `l` et un élément `x`
- renvoyant un booléen
- telle que `appartient l x` renvoie `true` si `x` apparait dans `l`, et `false` sinon

3.b) Recherche dans une liste triée

On suppose désormais que la liste est triée par ordre croissant.

Ecrivez une fonction `appartient2` :

- prenant en argument une liste `l` (supposée triée par ordre croissant) et un élément `x`
- renvoyant un booléen
- telle que `appartient2 l x` renvoie `true` si `x` apparait dans `l`, et `false` sinon

3.c) Insertion dans une liste triée

Ecrivez ensuite une fonction `insertion` :

- prenant en argument une liste `l` (supposée triée par ordre croissant) et un élément `x`
- renvoyant une liste
- telle que `appartient2 l x` renvoie une nouvelle liste triée par ordre croissant et composée des éléments de `l` et de l'élément `x`

3.d) Suppression dans une liste triée

Ecrivez enfin une fonction `suppression` :

- prenant en argument une liste `l` (supposée triée par ordre croissant) et un élément `x`
- renvoyant une liste
- telle que `appartient2 l x` renvoie une nouvelle liste triée par ordre croissant correspondant à la liste `l` où toutes les occurrences de `x` ont disparu.

Remarque : si `x` n'apparait pas `l`, la liste renvoyée est identique à `l`.

4) Somme et moyenne

4.a) Somme d'une liste d'entiers

Ecrivez une fonction `somme` :

- prenant en argument une liste d'entiers `l`
- renvoyant un entier
- telle que `somme l` renvoie la somme des éléments qui compose `l`.

4.b) Moyenne d'une liste de nombre réels

Ecrivez une fonction `moyenne` :

- prenant en argument une liste de nombres réels `l`
- renvoyant un nombre réel
- telle que `moyenne l` renvoie la moyenne des éléments qui compose `l`.

Variante ★ : Pouvez-vous trouver une méthode qui permettent de calculer cette moyenne en ne parcourant qu'une seule fois la liste ?

5) Tri fusion

Le *tri fusion* est l'un des tris classiques sur les listes. Basé sur le principe « Diviser pour régner », il repose sur l'observation suivante :

Pour trier une liste, il suffit de :

- *Couper cette liste en deux*
- *Trier ces deux moitiés indépendamment*
- *Fusionner les deux demi-listes triées en une seule liste triée*

Nous allons voir dans les questions suivantes comment réaliser ces différentes étapes.

5.a) Fusion de deux listes triées

Pour commencer, écrivez une fonction `fusion` :

- prenant en argument deux liste `l1` et `l2` (qu'on suppose triées)
- renvoyant une liste
- telle que `fusion l1 l2` renvoie la liste triée contenant exactement les éléments de `l1` et `l2`

5.b) Découpage d'une liste en deux moitiés de longueurs similaires

Ecrivez ensuite une fonction `decoupe` :

- prenant en argument une liste `l`
- renvoyant un couple de liste
- telle que `decoupe l` renvoie un couple `(l1, l2)` tel
    - La concaténation de `l1` et `l2` redonne la liste initiale `l`
    - Les longueurs de `l1` et `l2` différent d'au plus 1

Indice : Vous pouvez commencer par imaginer une fonction `premiere_moitie` qui renvoie la première moitié d'une liste

Remarque : pour renvoyer un couple de valeurs, utilisez des parenthèses : `(valeur1, valeur2)`

Variante ★ : Pouvez-vous trouver une méthode qui permette de découper ainsi une liste en ne parcourant qu'une seule fois cette dernière ?

5.c) Tri fusion

En vous aidant des fonctions `fusion` et `decoupe`, imaginez une fonction `tri_fusion` :

- prenant en argument une liste `l`
- renvoyant une nouvelle liste
- telle que `tri_fusion l` renvoie une liste `l2` triée par ordre croissant et contenant exactement les mêmes éléments que `l`

Indice 1 : Pensez récursif !

Indice 2 : Une fonction récursive a toujours un cas de base facile à résoudre. Quel est ce cas de base dans le cas du tri fusion ?