# TP n°2 - Tri d'une liste

**Q1** Pour générer des nombres aléatoires en OCaml :
- on initialise le générateur en appelant une fois la fonction `Random.self_init: unit -> unit`
- on obtient ensuite des entiers positifs pseudo-aléatoires utilisant la fonction `Random.int : int -> int` : `Random.int N` renvoie un entier entre 0 (inclus) et `N` (exclu).

Écrire une fonction `liste_alea int -> int list` prenant en argument un entier positif `n` et renvoyant une liste de `n` entiers aléatoires.

## 1 Tri par sélection

**Q2** Écrire une fonction `extrait_min : 'a list -> 'a * 'a list` qui prend en argument une liste et qui renvoie le couple constitué du plus petit élément de la liste et de la liste privée de son plus petit élément.

**Q3** En déduire une fonction `tri_selection : 'a list -> 'a list` qui trie une liste par ordre croissant.

**Q4** Quelle est la complexité de cet algorithme de tri ?

## 2 Tri par insertion

**Q5** Écrire une fonction `insere : 'a -> 'a list -> 'a list` qui prend en argument un élément et une liste supposée triée dans l'ordre croissant et qui insère l'élément à sa place dans la liste.

**Q6** Le tri par insertion consiste à trier récursivement la queue de la liste puis à insérer sa tête dans la liste triée obtenue. Écrire une fonction `tri_insertion : 'a list -> 'a list` qui réalise le tri par insertion d'une liste.

**Q7** Quelle est la complexité de cet algorithme de tri ?

**Q8** Écrire une version récursive terminale du tri par insertion.
On utilisera pour cela une fonction auxiliaire de type `'a list -> 'a list -> 'a list` dont le premier argument est une liste des éléments à insérer dans la liste déjà triée présente en deuxième argument.

## 3 Tri fusion

**Q9** Écrire une fonction `decoupe : 'a list -> 'a list * 'a list` qui prend une liste en argument et renvoie un couple de listes dont la concaténation contient exactement les éléments de la liste initiale, et dont la longueur diffère au maximum d'une unité.

**Q10** Écrire une fonction `fusionne : 'a list -> 'a list -> 'a list` prenant en argument deux listes supposées triées dans l'ordre croissant et les fusionnant en une liste triée.

**Q11** Le tri fusion consiste à découper une liste en deux, à trier récursivement chacune des deux listes, puis à les fusionner pour obtenir une liste triée. Écrire une fonction `tri_fusion : 'a list -> 'a list` qui réalise le tri fusion d'une liste.

*Cet algorithme a une complexité en $O(n \ln n)$ où $n$ est la longueur de la liste à trier.*

## 4 Tri rapide

**Q12** Écrire une fonction `partitionne : 'a -> 'a liste -> 'a liste * 'a liste` qui prend en argument un élément `x` et une liste `lst` et qui renvoie la liste des éléments de `lst` inférieurs ou égaux à `x` et la liste des éléments de `lst` strictement supérieurs à `x`.

**Q13** Le tri rapide consiste à isoler un élément de la liste à trier (par exemple le premier), à partitionner le reste de la liste en deux grâce à la fonction précédente, à trier récursivement les deux listes obtenues, puis à reconstruire la liste par concaténation. Écrire une fonction `tri_rapide : 'a list -> 'a list` qui réalise le tri rapide d'une liste.

*Cet algorithme a une complexité quadratique dans le pire des cas, mais une complexité en $O(n \ln n)$ en moyenne.*