TRIS

L'objectif de ce TP est d'étudier différents programmes pour trier une liste, c'est-à-dire d'obtenir les éléments rangés dans l'ordre croissant.
Commençons par quelques points de vocabulaire :

- La **clé du tri** est la relation d'ordre total sur les éléments de la liste à trier.
- Un **tri en place** lit et écrit directement dans la liste, et n'utilise que peu de mémoire par ailleurs.
- Un **tri stable** conserve dans le même ordre que dans la liste initiale les éléments qui ont la même clé.
- un **tri comparatif** se base sur des comparaisons entre les clés des éléments à trier.

1 Tri par sélection

On considère une liste `L` contenant n éléments. Le tri par sélection consiste à :

- rechercher le plus petit élément de la liste et le placer en première position ;
- rechercher le deuxième plus petit élément de la liste et le placer en deuxième position ;
- répéter ce procédé jusqu'à ce que tous les éléments aient été placés.

1. Écrire une fonction `minimum1` qui prend en argument une liste (non triée) et qui renvoie un couple formé du minimum et de sa position dans la liste.
2. Modifier la fonction précédente en une fonction `minimum2` prenant en argument, en plus de la liste, l'indice à partir duquel commencer la recherche.
3. En déduire une procédure `tri_selection` qui prend en argument une liste et qui la trie.
4. Quelles sont les caractéristiques du tri par sélection ?
5. Évaluer le nombre maximal de comparaisons effectuées pour trier une liste de taille n.

2 Tri par insertion

Le tri par insertion est celui habituellement utilisé pour trier ses cartes. Il consiste à insérer les éléments d'un partie de la liste non triée dans la liste triée. On détaille l'algorithme sur un exemple :
On considère la liste non triée `L = [8, 5, 3, 9, 2]` comprenant $n = 5$ éléments. On parcourt la liste du deuxième élément au dernier élément. Lorsqu'on est à l'étape k (k variant entre 1 et $n-1$) les éléments entre les indices 0 et $k-1$ sont triés, il faut donc insérer cet élément d'indice k dans la liste triée :

- Liste non triée : `L = [8, 5, 3, 9, 2]`
- $k = 1$, `L = [8, 5, 3, 9, 2]` $\longrightarrow$ `L = [ 5, 8, 3, 9, 2]` ;
- $k = 2$, `L = [5, 8, 3, 9, 2]` $\longrightarrow$ `L = [ 3, 5, 8, 9, 2]` ;
- $k = 3$, `L = [3, 5, 8, 9, 2]` $\longrightarrow$ `L = [ 3, 5, 8, 9, 2]` ;
- $k = 4$, `L = [3, 5, 8, 9, 2]` $\longrightarrow$ `L = [ 2, 3, 5, 8, 9]` ;

1. Écrire une fonction `tri_insertion` qui prend en argument une liste `L` et et la trie selon la méthode décrite dans l'exemple ci-dessus.
2. Donner les caractéristiques du tri par insertion.

3 Tri par partition-fusion

Le tri fusion est un algorithme récursif basé sur la principe de « diviser pour régner » :

- Si la liste est vide ou admet un seul élément alors elle est triée ;
- sinon, on divise la liste en deux moitiés, que l'on trie récursivement puis qu'on fusionne en les interclassant, de sorte à obtenir la liste de départ triée.

1. Écrire une fonction `fusion` qui prend en argument deux listes `L1` et `L2` triées et retourne la fusion triée des deux listes.
2. Écrire une fonction récursive `tri_fusion` implémentant le tri fusion.
3. Donner les caractéristiques de l'algorithme.

4 Tri rapide

Le tri rapide est également un algorithme récursif basé sur la principe de « diviser pour régner » :

- Si la liste est vide ou admet un seul élément alors elle est triée ;
- sinon :
  - on définit le **pivot** p comme le dernier élément de la liste.
  - On divise la liste en 3 : les éléments strictement inférieurs à p, ceux égaux à p, et ceux strictement supérieurs. On trie récursivement la première et la dernière liste, et on concatène les résultats.

1. Appliquer à la main l'algorithme à la liste `L = [10, 3, 5, 6, 8, 12, 4, 7]`.
2. Écrire une fonction `repartition` prenant en argument une liste et renvoyant les 3 listes décrites ci-dessus.
3. En déduire une implémentation du tri rapide.
4. Avec notre choix de pivot, que se passe-t-il si la liste est triée ? que pourrait-on faire pour éviter cela ?
5. On souhaite à présent écrire une implémentation en place du tri rapide.

(a) Écrire une fonction `repartition_en_place` prenant en argument une liste `L` ainsi que deux indices `a` et `b`, qui utilise `L[b]` comme pivot et permute les éléments de `L` entre ces indices de façon à ce qu'à gauche du pivot ne se trouve que des éléments qui lui sont inférieur, et à sa droite que des éléments qui lui sont strictement supérieurs. De plus la fonction renverra l'indice du pivot après permutation.

(b) Écrire une procédure `tri_sous_liste` prenant en argument une liste `L` et deux indices `a` et `b`, et qui trie en place selon la méthode du tri rapide la partie de `L` entre les indices `a` et `b` inclus.

(c) En déduire une procédure `tri_rapide_en_place` prenant en argument une liste `L` et la triant par un tri rapide en place.

5 Tri par comptage

Le tri par comptage est un algorithme de tri d'entiers. On considère des entiers de 0 à p dans une liste `L` contenant n éléments. Par exemple $p = 20$ et `L = [10, 20, 12, 12, 16, 16, 12, 20, 15]`. On relève les occurences de chacun des entiers compris entre 0 et p (ou m et M si on connait un minorant et un majorant des éléments à trier) dans une liste `D`, puis on remplit une autre liste `L1`, la liste triée, que l'on construit au fur et à mesure à partir de `D`.

1. Écrire une fonction `tri_comptage` qui prend en argument une liste d'entiers `L` et qui renvoie une nouvelle liste, correspondant à la liste `L` triée.
2. Quelles sont les caractéristiques du tri par comptage ?