■ T.P. 4 ■
Tris

1. Écrire une fonction `echange(liste,i,j)` qui échange, en la modifiant, les éléments d'indice `i` et `j` de la liste `liste`.

Partie I : Tri par sélection

Le tri par sélection consiste à mettre le plus petit élément de la liste en première position, puis à recommencer avec les éléments restants.

2. Écrire une fonction `mini(liste,n)` qui prend en argument une liste `liste` et un entier naturel non nul `n` et qui retourne le couple constitué du minimum des `n` derniers éléments de `liste` ainsi que l'indice de sa première apparition.

3. En utilisant la fonction `mini`, écrire une fonction `tri_selection(liste)` qui trie la liste `liste` selon la méthode de tri par sélection.

Partie II : Tri par insertion

Le tri par insertion consiste à garder le début de la liste triée et à y insérer successivement les éléments restants.

4. Écrire une fonction `tri_partiel(liste,n)` qui prend en argument une liste `liste` dont on suppose que les `n` premiers éléments sont triés et qui insère l'élément `liste[n]` à sa place parmi les `n` premiers éléments de `liste`.
Cette fonction utilisera la fonction `echange` *écrite dans le préambule.*

5. Écrire une fonction `tri_insertion(liste)` qui trie la liste `liste` selon la méthode du tri par insertion.

Partie III : Tri à bulles

Le tri à bulles consiste à parcourir la liste et à intervertir toutes les paires d'éléments consécutifs qui ne sont pas ordonnées. Le tri s'arrête lorsque toutes les paires sont bien ordonnées.

6. Écrire une fonction `tri_bulle(liste)` qui trie la liste `liste` selon la méthode du tri à bulles.

Partie IV : Application : Recherche dichotomique

On suppose dans cette partie que les listes passées en argument sont triées. Pour déterminer si l'élément x est dans la liste L de longueur n,

- On compare x à l'élément médian d'indice $m = \lfloor \frac{n}{2} \rfloor$.
  - Si $x = L[m]$, alors x est dans L.
  - Si $x < L[m]$, on recherche x dans la sous-liste $L[0 \dots m-1]$.
  - Si $x > L[m]$, on recherche x dans la sous-liste $L[m+1 \dots n-1]$.
- On s'arrête de chercher, soit dès que x est trouvé, soit dès que la recherche s'effectue dans $L[g \dots d]$ et que $g > d$.

7. Écrire une fonction `recherche_dicho(liste,x)` qui détermine si l'élément `x` est dans la liste `liste`.

Partie V : Pour aller plus loin : Tri par dénombrement

Dans cette partie, on considère une méthode de tri qui ne repose pas uniquement sur des comparaisons. On suppose que les éléments de la liste ne peuvent prendre qu'un nombre fini de valeurs, supposées être des entiers compris entre 0 et $m-1$ pour simplifier. Ce tri, appelé également tri du postier, peut être utilisé pour trier des enveloppes en fonction de l'adressage.

8. Écrire une fonction `denombrement(liste, m)` qui, étant donnée une liste `liste` d'entiers compris entre 0 et `m-1`, renvoie une liste `occ` de taille `m` telle que pour tout entier i, `occ[`i`]` soit égal au nombre d'occurrences de i.

9. Écrire une fonction `place(liste, m)` qui, étant donnée une liste `liste` d'entiers compris entre 0 et `m-1`, renvoie une liste `position` de taille `m` telle que pour tout entier i, `position[`i`]` soit la position du premier entier égal à i dans la liste triée.

10. Écrire une fonction `tri_postier(liste, m)` qui trie la liste en temps linéaire par rapport à sa longueur.