TP : Tris

Exercice 1 1. Ecrire une fonction `recherche_min(L)` qui prend en argument une liste de mots et renvoie l'indice du mot le plus court de la liste en nombre de lettres.

2. En s'inspirant de l'algorithme de tri par sélection, écrire un algorithme qui prend en argument une liste de mots et renvoie la lise triée du mot le plus court au mot le plus long.

3. Tester votre fonction avec la liste `['totot', 'bonjour', 'a', 'oui', 'non']`.

Exercice 2 *Temps d'exécution*
Pour mesurer le temps d'exécution d'un programme, on importe la fonction `time` avec l'instruction `from time import time`.
On souhaite comparer les temps d'exécution du tri par insertion et du tri fusion sur deux types de listes : une liste de nombres au hasard et une liste déjà triée.

1. Ecrire les algorithmes du tri par insertion et celui du tri fusion.

2. Construire une liste de 3000 entiers pris au hasard entre 1 et 10000, bornes comprises. Mesurer le temps d'exécution des deux algorithmes pour cette liste. On n'oubliera pas de reconstruire la liste entre les deux tris.

3. Quel commentaire peut on faire ?

4. Construire la liste des 3000 entiers de 0 à 2999. Mesurer le temps d'exécution des deux algorithmes pour cette liste.

5. Quel commentaire peut on faire ?

Exercice 3
L'objectif est d'écrire une programme qui trie une liste de mots (type `str`) et les range suivant l'ordre lexicographique (l'ordre des dictionnaires).

1. Ecrire la définition de la variable `alphabet` :

   ```
   alphabet = 'AaàBbCcDdEeéèFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz'
   ```

2. Ecrire une fonction `ordre_alphabetique` qui prend en caractère deux caractères alphabétiques `c1` et `c2` et renvoie -1 si `c1` est avant `c2`, 1 si `c2` est avant `c1` et 0 si `c1=c2`. On pourra utiliser la fonction `index` qui renvoie l'indice d'un élément dans une chaîne de caractères.

3. Ecrire une fonction `ordre_lexicographique` qui prend en argument deux mots `m1` et `m2` et renvoie -1 si « `m1 <<m2` »pour l'ordre lexicographique, 1 si « `m1 >>m2` »pour l'ordre lexicographique et 0 si « `m1 =m2` ». On utilisera la fonction `ordre_alphabetique`

4. Ecrire une fonction `tri_lexicographique` qui prend en argument une liste de mots et trie cette liste. On utilisera la fonction `ordre_lexicographique` avec l'algorithme du tri par insertion.

Exercice 4 *Trier des points*
On dispose de points dans le plan muni d'un repère orthonormé. Ces points possède des couples de coordonnées $(x;y)$ représentées par la liste `[x, y]`. Nous allons trier ces points en fonction de leur distance à l'origine $\mathcal{O}$ du repère, de la plus petite à la plus grande.

1. Ecrire une fonction `distance2` qui prend en paramètre une liste de deux nombres nommée `point` qui représente un point du plan et renvoie le carré de la distance de ce point à $\mathcal{O}$

2. Ecrire une fonction `compare` qui prend en paramètre deux ponts P_1 et P_2 et qui renvoie -1 si P_1 est plus proche de $\mathcal{O}$ que P_2, 1 si P_2 est plus proche de $\mathcal{O}$ que P_1 et 0 si les deux points sont équidistants de $\mathcal{O}$

3. Ecrire une fonction `tri_points` qui prend en paramètre une liste composée de listes de deux nombres représentant des points du plan et qui trie cette liste suivant la distance entre les points et $\mathcal{O}$. Utiliser un algorithme de tri du cours.