

EXERCICE 1 : Algorithmes de tri (4 points)

Cet exercice traite principalement du thème « algorithmique, langages et programmation ». Le but est de comparer le tri par insertion (l'un des algorithmes étudiés en 1^{ère} NSI pour trier un tableau) avec le tri fusion (un algorithme qui applique le principe de « diviser pour régner »).

Partie A : Manipulation d'une liste en Python

1. Donner les affichages obtenus après l'exécution du code Python suivant.

```
notes = [8, 7, 18, 14, 12, 9, 17, 3]
notes[3] = 16
print(len(notes))
print(notes)
```

2. Écrire un code Python permettant d'afficher les éléments d'indice 2 à 4 de la liste notes.

Partie B : Tri par insertion

Le tri par insertion est un algorithme efficace qui s'inspire de la façon dont on peut trier une poignée de cartes. On commence avec une seule carte dans la main gauche (les autres cartes sont en tas sur la table) puis on pioche la carte suivante et on l'insère au bon endroit dans la main gauche.

1. Voici une implémentation en Python de cet algorithme. Recopier et compléter les lignes 6 et 7 surlignées (uniquement celles-ci).

```
1 def tri_insertion(liste):
2     """ trie par insertion la liste en paramètre """
3     for indice_courant in range(1,len(liste)):
4         element_a_inserer = liste[indice_courant]
5         i = indice_courant - 1
6         while i >= 0 and liste[i] > ..... :
7             liste[.....] = liste[.....]
8             i = i - 1
9         liste[i + 1] = element_a_inserer
```

On a écrit dans la console les instructions suivantes :

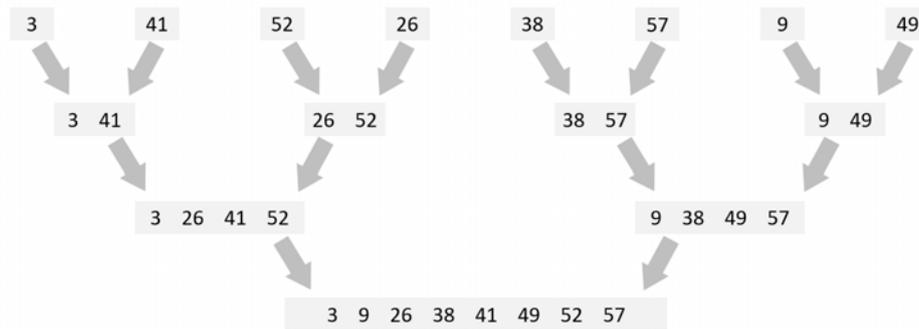
```
notes = [8, 7, 18, 14, 12, 9, 17, 3]
tri_insertion(notes)
print(notes)
```

On a obtenu l'affichage suivant : [3, 7, 8, 9, 12, 14, 17, 18]

- Expliquer le rôle de la première ligne du code de la question 3.

Partie D : Comparaison du tri par insertion et du tri fusion

Voici une illustration des étapes d'un tri effectué sur la liste [3, 41, 52, 26, 38, 57, 9, 49].



- Quel algorithme a été utilisé : le tri par insertion ou le tri fusion ? Justifier.
- Identifier le tri qui a une complexité, dans le pire des cas, en $O(n^2)$ et identifier le tri qui a une complexité, dans le pire des cas, en $O(n \log_2 n)$.
Remarque : n représente la longueur de la liste à trier.
- Justifier brièvement ces deux complexités.