Algorithmie
Épreuve pratique
Tri insertion

1 Spécifications

1.1 Spécifications algorithme

Ecrire en pseudo code une fonction nommée **tri_ insertion(T)** qui prend en argument un tableau T et qui renvoie le tableau T trié en place.
La méthode utilisée pour trier est la méthode dite du **tri insertion** :
On commence avec une liste déjà triée vide. On itère sur la liste et, à chaque tour on insère le premier élément de la liste restante afin que la première partie de la liste soit triée.

1.2 Spécifications programme Python

Ecrire en langage Python une fonction nommée **tri_ insertion(T)** qui prend en argument une liste Python T et qui renvoie le tableau T trié en place.
La méthode utilisée pour trier est la méthode dite du **tri insertion** :
On commence avec une liste déjà triée vide. On itère sur la liste et, à chaque tour on insère le premier élément de la liste restante afin que la première partie de la liste soit triée.

2 Résolution

On insère un à un les éléments parmi ceux déjà trié.
☞ Le ***tri par insertion*** est aussi appelé "tri naturel" : c'est souvent ce tri que nous utilisons naturellement pour trier un jeu de cartes par exemple.

VIDÉO Une animation pour comprendre le tri par insertion - Cliquez ici !

2.1 Un algorithme

```
1 FONCTION tri_ insertion(T :tableau d'entiers)
2 |  POUR i DE 1 A longueur(T) − 1 FAIRE
3 |  |  x ← T[i]
4 |  |  j ← i
5 |  |  TANT QUE j > 0 and T[j − 1] > x FAIRE
6 |  |  |  t[j] ← t[j − 1]
7 |  |  |  j ← j − 1
8 |  |  t[j] ← x
```

2.2 Une implémentation en Python

```
for i in range(1,len(T)):
    x = T[i]
    j = i
    while j > 0 and T[j-1] > x:
        t[j] = T[j-1]
        j = j - 1
    T[j] = x
```

⚠ Il faut être capable de :

- Documenter la fonction
- Ecrire les pré-conditions
- Ecrire les post-conditions
- Donner un jeu de tests pertinents
- Utiliser des asserts pour vérifier ces tests
- connaître la complexité temporelle
- Prouver la terminaison de l'algorithme

2.3 Complexité temporelle

Propriété 0.1

La complexité temporelle d'un tri insertion est, dans le pire des cas, en $\mathcal{O}(n^2)$. On parle de complexité quadratique.

2.4 Terminaison

2.5 Terminaison

La boucle POUR ne pose aucun problème !
En revanche, il faut faire attention à la boucle TANT QUE qui peut ne jamais terminer !
La condition de la boucle TANT QUE "est $j < 0$ and $T[j-1] > x$".
La boucle s'arrêtte quant l'une ou l'autre des conditions n'est plus vraies.
Ici, on décrémente j à chaque fois de, donc nous sommes certains que la condition $j > 0$ ne sera plus vraie à partir d'une certaine étape !

☞ Notre algorithme s'arrêtera donc !

3 Questions possibles

☛ Modifier le programme pour qu'il ne modifie plus en place le tableau entré en argument, mais renvoie un nouveau tableau trié.