Exercice 1

On rappelle ci-dessous l'algorithme du Tri_insertion appliqué à un tableau de n nombres .

Tri_insertion(A[1..n])
Entrée : Un tableau de n nombres $[a_1, .., a_n]$
Sortie : le tableau trié dans l'ordre croissant

```
1  n ← taille du tableau A
1  Pour j allant de 2 a n :
2     cle ← A[j]
3     // Insere A[j] dans la suite triee A[1,..,j-1]
4     i ←j-1
5     Tant que i >0 et A[i]>cle Faire :
6           A[i+1]← A[i]
7           i← i-1
8     A[i+1] ←cle
9  Renvoyer A
```

1. Appliquer l'algorithme au tableau de nombres : [15 , 8 , 19 , 10 , 16, 5]

   On écrira le contenu du tableau à la fin de chaque itération de la boucle principale **Pour** (ligne 1).
   On complètera le tableau ci-dessous .

Tableau initial	15	8	19	10	16	5
fin itération 1						
fin itération 2						
fin itération 3						
fin itération 4						
.....						
.....						
.....						

   Réponse:On compte les comparaisons entre éléments du tableau

Tableau initial	15	8	19	10	16	5	1 comparaison
fin itération 1	8	15	19	10	16	5	1 comparaison
fin itération 2	8	15	19	10	16	5	3 comparaisons
fin itération 3	8	10	15	19	16	5	2 comparaisons
fin itération 4	8	10	15	16	19	5	5 comparaisons
fin itération 5	4	5	8	10	15	19	
.....							
.....							

2. Combien y a-t-il de comparaisons (boucle Tant Que , ligne 5)? **Réponse:** 12 comparaisons

3. Ecrire en langage Python la fonction `tri_insertion`(tab) où `tab` est une liste contenant des entiers.

```
def tri_inserton(A):
    n=len(A)
    for j in range(1,n):
        cle = A[j]
        i = j - 1
        while i>=0 and A[i] > cle :
            A[i+1] = A[i]
            i = i - 1
        A[i+1] = cle
    return A
```

4. Modifier la fonction `tri_insertion` pour qu'elle renvoie le nombre de comparaisons effectuées sur les éléments du tableau tab.

```
def tri_inserton(A):
    n=len(A)
    nombre_comparaisons = 0
    for j in range(1,n):
        cle = A[j]
        i = j - 1
        while i>=0 and A[i] > cle :
            nombre_comparaisons += 1
            A[i+1] = A[i]
            i = i - 1
        A[i+1] = cle
    return nombre_comparaisons
```

5. Quelle est la complexité de la fonction `tri_insertion` dans le pire des cas?

   Réponse : $O(n^2)$

Exercice 2

Rappels : Une adresse IPv4 est composée de 4 octets, soit 32 bits. Elle est notée A.B.C.D où A, B, C et D représentent les 4 octets de l'adresse. La notation A.B.C.D/n signifie que les n premiers bits de l'adresse IP représentent la partie "réseau", les bits qui suivent représentent la partie "hôte". Les n premiers bits du masque sont donc égaux à 1 et les autres à 0.

L'adresse IPv4 dont tous les bits de la partie "hôte" sont à 0 est appelée "adresse du réseau". L'adresse IPv4 dont tous les bits de la partie "hôte" sont à 1 est appelée "adresse de diffusion".

Le poste PC LG 03 de la salle informatique du Lycée général a pour adresse IPV4 : 192.168.162.4. On donne ci-dessous le début de l'écriture binaire de cette adresse.

1. Recopier et compléter cette adresse IP : 11000000.xxxxxxxx.xxxxxxxx.xxxxxxxx

   Réponse: **11000000.10101000.10100010.0000100**

2. Le poste PC Admin 02 du secteur "Administration" a pour adresse IPv4 : 192.168.16.12/24.

   (a) Donner l'adresse du réseau local dédié au secteur "Administration" et son masque de sous-réseau.
   Réponse: les 24 premiers bits sont réservés à la partie réseau,d'où**192.168.16.0**

   (b) Donner l'adresse de diffusion (appelée aussi broadcast) de ce réseau.
   Réponse : **192.168.16.255** (les bits de la partie hôte sont tous à 1)

   (c) Donner le nombre maximal de machines que l'on peut connecter sur ce réseau.
   Réponse: il y a 256 possibilités sur le dernier octet.On enlève l'adresse du réseau et le broadcast.Il reste 254 machines qu'on peut brancher sur le réseau.

   (d) On veut ajouter un nouvel ordinateur au réseau dédié au secteur "Administration".Donner à cet ordinateur une adresse IP valable.
   Réponse: par exemple 192.168.16.13