Exercices 9

1. Si la liste L contient N = 1000 éléments, combien de pas sont effectués dans la fonction suivante? Autrement dit, combien de fois le test L[i]==L[j] est exécuté ?

```
def ma_fonction(L):
    compteur=0
    for i in range(len(L)):
        semaphore=True
        for j in range(len(L)):
            if L[i] == L[j] and i!=j:
                semaphore = False
        if semaphore:
            compteur=compteur+1
    return compteur

L=[1, 0, 2, 0, 3]
resultat = ma_fonction(L)
print(resultat)
```

Réponses possibles (choisir une):

a) Moins de 20
b) Entre 1 000 et 9 000 fois
c) Entre 10 000 and 100 000 fois
d) Entre 200 000 et 2 millions de fois
e) Entre 10 millions et 200 millions de fois

2. Que fait la fonction de la question antérieure? Quelle est la description la plus correcte de la fonction?

Réponses possibles (choisir une):

a) Compte combien d’éléments arrivent seulement une fois dans la liste.
b) Compte combien d‘éléments arrivent au moins deux fois dans la liste.
c) Vérifie s’il y a des éléments qui arrivent au moins deux fois dans la liste
d) Vérifie s‘il y a des éléments qui arrivent seulement une fois dans la liste.
e) Aucune des réponses ci-dessus.

3. Pour la fonction suivante, combien d'opérations print sont exécutées dans la fonction, approximativement?

```
def f(n):
  for i in range(n) :
     for j in range(n):
        for k in range(n):
           print('*')
```

a) n
b) n^2
c) n^3
d) log(n)

Autres exercises:

L'ordre de complexité est utilisé pour mesurer le nombre des opérations exécutées par un algorithme dans le pire des cas (en anglais: Big O notation). Les constantes et les termes plus petits sont ignorés quand n deviens large. Par exemple:
$4n^2$ est $O(n^2)$
$n^2/100 + n$ est $O(n^2)$
$20n - 70$ est $O(n)$
$10 \log_2 n + 100$ est $O(\log_2 n)$

Pour chacune des 7 fonctions suivantes, combien d'opérations sont exécutées dans la fonction, approximativement, quand n devines large? Par exemple, pour f3, la réponse est: approximativement linéaire avec n, donc O(n).

```
def f2(n):
    i = n
    while i >= 1 :
        i = i // 2

def f3(n):
  for i in range(n) :
    print('*')

def f4(n):
  for i in range(-n,n,3) :
    print('*')

def f5(n):
  for i in range(n) :
      for j in range(i+1, n):
         print('*')
```