TD1

Exercice 1.
Pour chacun des algorithmes suivants, donnez la valeur de c en fonction de n à la fin de l'algorithme puis la complexité de l'algorithme.

```
Algo 1 :
    c <- 0
    Pour i allant de 1 à n:
        c <- c + 1

Algo 2 :
    c <- 0
    Pour i allant de 1 à n:
        c <- c + 1
    Pour j allant de 2 à  n+1:
        c <- c + 1

Algo 3 :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à n:
            c <- c + 1

Algo 4 :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à i:
            c <- c + 1
.
```

```
Algo 5 :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à n:
            c <- c + 1
    Pour k allant de 1 à n:
        c <- c + 1

Algo 6 :
    c <- 0
    Pour i allant de 1 à n:
        c <- c + 1
        Pour j allant de 1 à n:
            c <- c + 1

Algo 7 :
    c <- 0
    i <- 1
    Tant que i < n:
        i <- i+2
        c <- c + 1

Algo 8 :
    c <- 0
    i <- 1
    Tant que i < n:
        i <- i*2
        c <- c + 1
```

Exercice 2.
On considère des algorithmes de complexité : $\log(n)$, $\sqrt{n}$, n, $50 * n$, $n\log(n)$, n^2, n^3, 2^n.

(1) Calculez chacune de ces fonctions pour les puissances de 10 de 10^1 à 10^{10}. Le résultat sera sera donné sous la forme approchée $x.10^k$ (Pour 2^n one ne calculera que jusqu'à $n = 10^3$).

(2) En supposant que l'on dispose d'une machine capable de faire 10^6 (1 million) d'opérations par seconde, quelle est la taille des problèmes que l'on peut résoudre en une seconde, 1000 secondes (environ 17 minutes), 10 000 secondes (environ 2h47) ?

Exercice 3.
Un nombre premier est un nombre qui n'a pas de diviseurs autre que lui-même et 1. Écrivez un algorithme testant la primalité d'un nombre, la complexité doit être inférieure à $O(n)$ (sous-linéaire).

Exercice 4.
Écrivez un algorithme qui prend en paramètre un tableau d'entiers de taille n *triés* (les valeurs sont dans l'ordre croissant) ainsi qu'un entier a et détermine si a est dans T. Quelle est la complexité de votre algorithme ?

Exercice 5.
Une matrice carrée de taille n est un double tableau de taille $n \times n$. Soient A et B deux matrices de taille n dont on écrit les entrées $a_{i,j} = A[i][j]$ et $b_{i,j} = B[i][j]$ avec $0 \leq i, j < n$. Le produit $A \times B$ est donné par la matrice C dont les entrées sont calculées par la formule

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}.$$

(1) Écrivez l'algorithme qui calcule la matrice C en fonction de A et B (on supposera qu'on possède une fonction CreeMatrice(n) qui crée une matrice carrée de taille n et l'initialise à 0.

(2) Quelle est la complexité de votre algorithme ? La fonction CreeMatrice peut-elle l'influencer ?