---

**Exercice 1**. O(f(n)) : Montrer que la relation « est de l'ordre de » est une relation réflexive, transitive, mais pas symétrique.

---

**Exercice 2**. Trouver l'erreur

a) $O(n) = O(n^3 + (n - n^3)) = O(\max\{n^3, n - n^3\}) = O(n^3)$
b) $O(n^2) = O(n + n + \ldots +n) = O(\max\{n , \ldots, n\}) = O(n)$

---

**Exercice 3**. Parmi les assertions suivantes, lesquelles sont vraies ?

a) $3^n \in O(2^n)$
b) $2^n \in O(3^n)$
c) $2^n \in O(n!)$
d) $n! \in O(2^n)$
e) $n! \in O(2^{n\log(n)})$
f) $5n \in O(2n)$
g) $n^2 \in O(10^{-5}\, n^3)$
h) $25n^4 - 19\, n^3 + 13n^2 \in O(n^4)$
f) $2^{n+100} \in O(2^n)$
j) $k \in O(1)$ (k est une constante)

---

**Exercice 4**. Calculer la complexité de l'algorithme suivant :

```
a := 1
Pour i := 1 à x faire
        k := 0
        Pour j := 1 à 2a faire
                k := k+1
        finpour
        a := k
finpour
```

---

**Exercice 5**. Soit A un tableau de n entiers ($n \geq 1$).

a) Ecrire un algorithme itératif qui calcule la somme des éléments de A et prouver cet algorithme. Déterminer sa complexité.
b) Ecrire un algorithme récursif qui calcule la somme des éléments de A et prouver cet algorithme. Déterminer sa complexité.

---

**Exercice 6**. Soit la fonction récursive F d'un paramètre entier n suivante :

```
Fonction F(n : entier) : entier
Si n = 0        alors retourner (2)
                Sinon retourner (F(n-1) * F(n-1))
finsi
```

a) Que calcule cette fonction ? le prouver.
b) Déterminer la complexité de la fonction F. Comment améliorer cette complexité ?

Soit la fonction G suivante :

**Fonction** G(n : entier) : entier
R : entier
**Pour** i := 1 à n **faire** R := R * R **finpour**
**Retourner** (R)

c) Que calcule cette fonction ? le prouver.
d) Déterminer la complexité de la fonction G.

---

**Exercice 7**. Déterminer la complexité des algorithmes suivants par rapport au nombre d'itérations effectuées où m et n sont deux entiers positifs :

**Algorithme A**
i:=1 ; j:=1 ;
**Tantque** (i≤m) et (j≤n) **faire** i:=i+1 ; j:=j+1 **fintantque**

**Algorithme B**
i:=1 ; j:=1 ;
**Tantque** (i≤m) ou (j≤n) **faire** i:=i+1 ; j:=j+1 **fintantque**

**Algorithme C**
i:=1 ; j:=1 ;
**Tantque** (j≤n) **faire Si** (i≤m) **alors** i:=i+1 **sinon** j:=j+1 **finsi fintantque**

**Algorithme D**
i:=1 ; j:=1 ;
**Tantque** (j≤n) **faire Si** (i≤m) **alors** i:=i+1 **sinon** j:=j+1 ; i:=1 **finsi fintantque**

---

**Exercice 8.** Ecrire en C une fonction récursive permettant de calculer $x^N$ (N entier positif et X un réel) en utilisant les relations suivantes où N/2 désigne la division entière :

a) $X^N = 1$ si $N = 0$
$X^N = X * X^{N-1}$ si $N > 0$
b) $X^N = 1$ si $N = 0$
$X^N = (X * X)^{N/2}$ si $N > 0$ et N est pair
$X^N = X * (X * X)^{N/2}$ si $N > 0$ et N est impair

Comparer les complexités temporelles de ces deux versions (a) et b)).