UNIVERSITÉ DE DOUALA
ÉCOLE SUPÉRIEURE TECHNIQUE LA SALLE
Algorithmique

Année académique 2026-2027
Classe : **IE4**
Pr Moffo Lonla Bertrand

Travaux Dirigés n° 3

Structures décisionnelles, boucles et complexité

Exercice 1 : Le plus grand de trois nombres

Écrire un algorithme qui affiche le plus grand de trois nombres réels saisis au clavier.

Exercice 2 : Équation du second degré

Écrire un algorithme qui résout $ax^2+bx+c=0$ dans $\mathbb{R}$ (a, b, c réels saisis). Traiter le cas $a=0$.

Exercice 3 : Menu avec Selon

Écrire un algorithme qui lit un numéro de jour (1 à 7) et affiche le nom du jour (1 = lundi). Un numéro hors de cet intervalle provoque le message « Jour invalide ».

Exercice 4 : Boucles classiques

1. Calculer la factorielle $n!$ d'un entier $n \geq 0$ saisi, avec une boucle **Pour**.
2. Afficher les nombres pairs compris entre 0 et N (N saisi).
3. Calculer le PGCD de deux entiers strictement positifs par soustractions successives (tant que $a \neq b$, on remplace le plus grand par la différence).
4. Lire un entier n en contrôlant qu'il est strictement positif, puis afficher la somme de ses chiffres.

Exercice 5 : Simulation et rôle d'une boucle

Simuler l'algorithme suivant avec la suite de saisies `4, 7, 10, 3, -1`. Que réalise-t-il ? Que se passe-t-il si la première valeur saisie est -1 ? Corriger.

```
Algorithme : Nombres
Variables
    x, N, Np : entier
    pourcent : réel
Début
    N ← 0
    Np ← 0
    Répéter
        Lire(x)
        N ← N + 1
        Si x mod 2 = 0 alors
            Np ← Np + 1
        FinSi
    Jusqu'à x = -1
    pourcent ← Np * 100 / N
    Écrire(Np, " pairs, soit ", pourcent, " %")
Fin
```

Exercice 6 : Complexité

Donner, en fonction de n, le nombre d'exécutions de l'instruction `c ← c + 1` puis la complexité en notation O de chaque fragment.

1. **Pour** i ← 1 **à** n **faire Pour** j ← 1 **à** n **faire** c ← c + 1
2. **Pour** i ← 1 **à** n **faire Pour** j ← 1 **à** i **faire** c ← c + 1
3. **Pour** i ← 1 **à** n **faire** c ← c + 1 ; **Pour** j ← 1 **à** n **faire** c ← c + 1
4. i ← n ; **TantQue** i > 1 **faire** c ← c + 1 ; i ← i div 2