Correction - Algorithmique et Mathématiques

Auteurs : N'jie ZAMON, Joachim MARIE-LUCE

Partie 1 : Algorithmique

Exercice 1 : Tri d'une liste

Correction :
Voici une solution possible pour trier une liste de nombres en utilisant l'algorithme de tri à bulles :

```
def tri_liste(liste):
    n = len(liste)
    for i in range(n):
        for j in range(0, n-i-1):
            if liste[j] > liste[j+1]:
                liste[j], liste[j+1] = liste[j+1], liste[j]
    return liste

# Exemple d'utilisation
print(tri_liste([5, 2, 9, 1, 5, 6]))  # Retourne [1, 2, 5, 5, 6, 9]
```

Exercice 2 : Explication d'un algorithme

Correction :
Le programme définit une fonction `divise` qui calcule le quotient de la division d'un entier `dividende` par un entier `diviseur` sans utiliser l'opérateur de division.

Voici le fonctionnement détaillé :

— Si le `diviseur` est égal à zéro, la fonction retourne "Impossible !" car la division par zéro n'est pas définie.
— Sinon, la fonction initialise le `quotient` à zéro.
— Ensuite, elle entre dans une boucle qui continue tant que `dividende` est supérieur ou égal à `diviseur`.

— Dans la boucle, le `dividende` est réduit de la valeur du `diviseur`, et le `quotient` est incrémenté de 1 à chaque itération.
— Lorsque la boucle se termine, la fonction retourne le `quotient` calculé.

Exercice 3 : Calcul de la factorielle

Correction :
Voici une fonction pour calculer la factorielle d'un nombre entier positif :

```
def factorielle(n):
    resultat = 1
    for i in range(1, n + 1):
        resultat *= i
    return resultat

# Exemple d'utilisation
print(factorielle(5))  # Retourne 120
```

Une autre méthode pour calculer la factorielle serait d'utiliser la récursivité :

```
def factorielle(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorielle(n - 1)

# Exemple d'utilisation
print(factorielle(5))  # Retourne 120
```

Partie 2 : Mathématiques

Exercice 1 : Équations

Correction :

Résolvons les équations une par une.

a) $3x^2 + 2x - 1 = 0$

Pour résoudre cette équation quadratique, nous utilisons la formule quadratique :

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ici, $a = 3$, $b = 2$, $c = -1$.

$$x = \frac{-2 \pm \sqrt{(2)^2 - 4 \times 3 \times (-1)}}{2 \times 3} = \frac{-2 \pm \sqrt{4 + 12}}{6} = \frac{-2 \pm \sqrt{16}}{6} = \frac{-2 \pm 4}{6}$$

Donc les solutions sont :

$$x_1 = \frac{2}{6} = \frac{1}{3} \quad \text{et} \quad x_2 = \frac{-6}{6} = -1$$

b) $e^{2x} = 7$

Prenons le logarithme naturel des deux côtés :

$$2x = \ln(7) \implies x = \frac{\ln(7)}{2}$$

c) $\log_2(x - 1) = 4$

En utilisant la définition des logarithmes, nous avons :

$$x - 1 = 2^4 \implies x - 1 = 16 \implies x = 17$$

Exercice 2 : Multiplication

Correction :

Voici une fonction Python qui calcule le produit de deux nombres sans utiliser l'opérateur de multiplication et sans utiliser la fonction `abs()` :

```
def multiplication(a, b):
    resultat = 0
    positif_a = a if a >= 0 else -a  # Prendre a positif
    positif_b = b if b >= 0 else -b  # Prendre b positif
```

```
    for _ in range(positif_b):
        resultat += positif_a

    if (a < 0) ^ (b < 0):  # Vérifie si un des nombres est négatif
        resultat = -resultat

    return resultat

# Exemple d'utilisation
print(multiplication(3, 4))   # Retourne 12
print(multiplication(-3, 4))  # Retourne -12
print(multiplication(3, -4))  # Retourne -12
print(multiplication(-3, -4)) # Retourne 12
```

Exercice 3 : PGCD

Correction :
Voici une fonction Python pour calculer le PGCD de deux nombres en utilisant l'algorithme d'Euclide :

```
def PGCD(a, b):
    while b != 0:
        r = a - (a // b) * b  # Calcul du reste sans utiliser %
        a, b = b, r
    return a

# Exemple d'utilisation
print(PGCD(18, 24))  # Retourne 6
print(PGCD(42, 56))  # Retourne 14
```