EXERCICE 1 (4 points)

L'opérateur « ou exclusif » entre deux bits renvoie 0 si les deux bits sont égaux et 1 s'ils sont différents. Il est symbolisé par le caractère $\oplus$.
Ainsi :

- $0 \oplus 0 = 0$
- $0 \oplus 1 = 1$
- $1 \oplus 0 = 1$
- $1 \oplus 1 = 0$

On représente ici une suite de bits par un tableau contenant des 0 et des 1.

Exemples :

```
a = [1, 0, 1, 0, 1, 1, 0, 1]
b = [0, 1, 1, 1, 0, 1, 0, 0]
c = [1, 1, 0, 1]
d = [0, 0, 1, 1]
```

Écrire la fonction `ou_exclusif` qui prend en paramètres deux tableaux de même longueur et qui renvoie un tableau où l'élément situé à position `i` est le résultat, par l'opérateur « ou exclusif », des éléments à la position `i` des tableaux passés en paramètres.

En considérant les quatre exemples ci-dessus, cette fonction donne :

```
>>> ou_exclusif(a, b)
[1, 1, 0, 1, 1, 0, 0, 1]
>>> ou_exclusif(c, d)
[1, 1, 1, 0]
```

EXERCICE 2 (4 points)

Dans cet exercice, on appelle carré d'ordre n un tableau de n lignes et n colonnes dont chaque case contient un entier naturel.

Exemples :

1	7
7	1

c2

Un carré d'ordre 2

3	4	5
4	4	4
5	4	3

c3

Un carré d'ordre 3

2	9	4
7	0	3
6	1	8

c3bis

Un autre carré d'ordre 3

Un carré est dit semimagique lorsque les sommes des éléments situés sur chaque ligne et chaque colonne sont égales.

- Ainsi c2 et c3 sont semimagiques car la somme de chaque ligne et chaque colonne est égale à 8 pour c2 et 12 pour c3.
- Le carré c3bis n'est pas semimagique car la somme de la première ligne est égale à 15 alors que celle de la deuxième ligne est égale à 10.

La classe `Carre` en page suivante contient des méthodes qui permettent de manipuler des carrés :

- La méthode constructeur crée un carré sous forme d'un tableau à deux dimensions à partir d'une liste d'entiers, et d'un ordre.
- La méthode `affiche` permet d'afficher le carré créé.

Exemple :

```
>>> liste = (3, 4, 5, 4, 4, 4, 5, 4, 3)

>>> c3 = Carre(liste, 3)

>>> c3.affiche()
[3, 4, 5]
[4, 4, 4]
[5, 4, 3]
```

Compléter la méthode `est_semimagique` qui renvoie `True` si le carré est semimagique, `False` sinon. Puis tester la fonction `est_semimagique` sur les carrés c2, c3 et c3bis.

```
class Carre:
    def __init__(self, liste, n):
        self.ordre = n
        self.tableau = [[liste[i + j * n] for i in range(n)] for
j in range(n)]

    def affiche(self):
        '''Affiche un carré'''
        for i in range(self.ordre):
            print(self.tableau[i])

    def somme_ligne(self, i):
        '''Calcule la somme des valeurs de la ligne i'''
        somme = 0
        for j in range(self.ordre):
            somme = somme + self.tableau[i][j]
        return somme

    def somme_col(self, j):
        '''Calcule la somme des valeurs de la colonne j'''
        somme = 0
        for i in range(self.ordre):
            somme = somme + self.tableau[i][j]
        return somme

    def est_semimagique(self):
        s = self.somme_ligne(0)

        #test de la somme de chaque ligne
        for i in range(...):
            if ... != s:
                return ...

        #test de la somme de chaque colonne
        for j in range(...):
            if ... != s:
                return ...

        return ...
```