Proposition de correction

Exercice 1

Q1

Pile gauche : '7','8','9','10'

Pile droite : 'V','D','R','A'

mélange : ['10', 'A', '9', 'R', '8', 'D', '7', 'V']

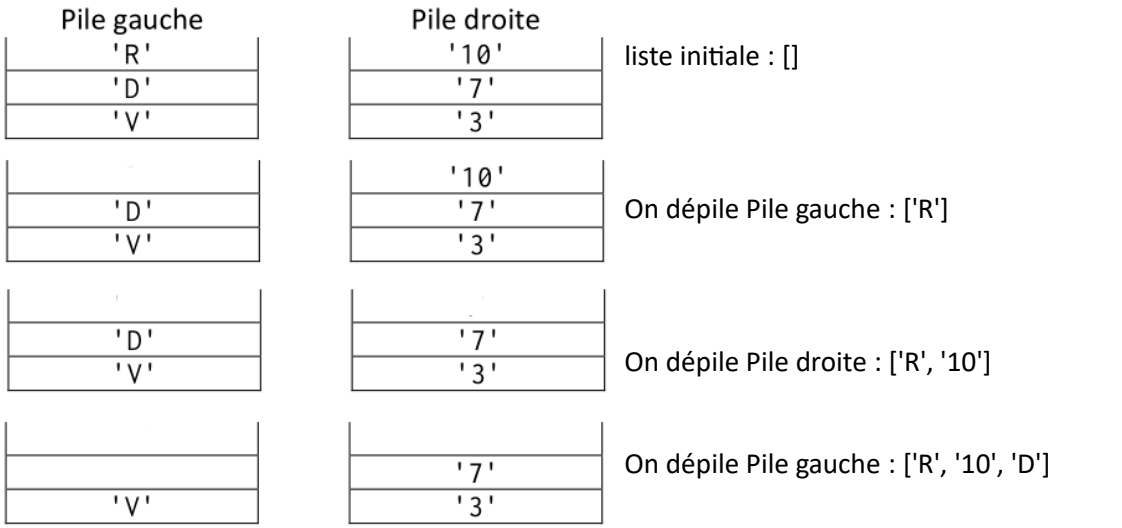
Q2

```
def liste_vers_pile(L):
    """prend en paramètre une liste et renvoie une pile"""
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        p_temp.empiler(L[i])
    return p_temp
```

Q3

3
2
1
6
5
4

Q4a



Q4b

```
def fusion(p1 : object, p2 : object) -> list:
    """ renvoie une liste construite à partir des deux piles p1 et p2."""
    liste = []
    p1_temp = p1.copier()
    p2_temp = p2.copier()
    while not p1_temp.est_vide() and not p2_temp.est_vide():
        liste += [p1_temp.depiler()] + [p2_temp.depiler()]
    return liste
```

Q5

```
def affichage_pile(p):
    p_temp = p.copier()
    if p_temp.est_vide():
        print('____')
    else:
        elt = p_temp.depiler()
        print('| ', elt, ' |')
        affichage_pile(p_temp)
```

Exercice 2

Q1

```
def mur(labyrinthe : list, i : int, j : int) -> bool:
    """renvoie un booléen indiquant la présence ou non d'un mur."""
    if len(labyrinthe) == 0:
        return None
    if 0 <= i < len(labyrinthe) and 0 <= j < len(labyrinthe[0]):
        return labyrinthe[i][j] == 1
```

Q2a

La distance entre 2 points voisins doit être égale à 1. On ne considère pas les cases en diagonales (distance > 1).

Donc les distances -1 ≤ l1-l2 ≤ 1 et -1 ≤ c1-c2 ≤ 1. Le carré permet de simplifier les tests sur le signe.

Q2b

```
def adjacentes(cases : list) -> bool:
    """ indique si la liste des cases forme une chaîne de cases adjacentes. """
    if len(cases) > 1:
        courant = cases[0]
        for i in range(1, len(cases)):
            if not voisine(courant, cases[i]):
                return False
            courant = cases[i]
        return True
    return False
```

Q3

Dans le pire des cas : pas de mur, le booléen possible est toujours vrai.

Cependant, i → +∞ et deviendra > à la taille du tableau.

Une des conditions n'étant plus vérifiée la condition booléenne de continuation de la boucle Tant que avec la fonction ET devient fausse.

Q4

```
def echappe(cases : list, laby : list) -> bool:
    """ indique si le chemin cases permet d'aller de l'entrée à la sortie du labyrinthe laby."""
    i, j = cases[0]
    if 0 < i < len(laby) - 1 and 0 < j < len(laby[0]) - 1:   # la case de départ doit être sur un bord
        return False
    i, j = cases[len(cases) - 1]
    if 0 < i < len(laby) - 1 and 0 < j < len(laby[0]) - 1:   # idem pour la case d'arrivée
```

```
            return False

        return teste(cases, laby)
```

Exercice 3

Q1

89 = 64 + 25 = 64 + 16 + 9 = 64 + 16 + 8 + 1 =%0101 1001

Q2

```
  11001110
⊕ 01101011
----------
= 10100101
```

Q3

```
def xor_crypt(message : str, cle : str) -> list:
    """ renvoie la liste des entiers correspondant au message crypté.
    pré : message à crypter et clé de cryptage de même longueur"""
    crypte = []
    if len(message) == len(cle):
        for i in range(len(message)):
            crypte.append( xor(ord(message[i]), ord(cle[i])) )
    return crypte
```

Q4

```
def generer_cle(mot : str, n : int) -> str:
    """ renvoie la clé de longueur n à partir de la chaîne de caractères mot."""
    cle = ""
    if len(mot) and n > 0:
        i = 0
        while i < n:
            cle += mot[i % len(mot)]    # la clé peut être plus longue que n
            i += 1
    return cle
```

Q5

$E1$	$E2$	$E1 \oplus E2$	$(E1 \oplus E2) \oplus E2$
0	0	0	0
0	1	1	0

1	0	1	1
1	1	0	1

Pour déchiffrer un message il faut appliquer de nouveau la fonction XOR au message chiffré avec la clé de chiffrement.

Exercice 4

Q1a

Non, car il peut y avoir des homonymes.

Q1b

id_licencie

Q2a

renvoie tous les prenom et nom des licenciés de l'équipe des moins de 12 ans.

Q2b

renvoie tous les attributs de la table licenciés de l'équipe des moins de 12 ans.

Q2c

SELECT date FROM matchs WHERE lieu = 'Domicile' AND equipe = 'Vétérans'

Q3

INSERT INTO licencies VALUES(287, 'Jean', 'Lavenu', 2001, 'Hommes 2')

Q4

UPDATE licencies SET equipe = 'Vétérans' WHERE prenom = 'Joseph' AND nom = 'Cuviller'

Q5

SELECT nom FROM licencies
JOIN Matchs ON licencies.equipe = matchs.equipe
WHERE matchs.adversaire = 'LSC' AND matchs.date = '2021-06-19'

Exercice 5

Q1a

La LED1 est bleue → (0,0, 255)

Q1b

16711680

Q1c

- coul = Obj_bandeau.get_pixel_rgb(0) # récupère le tuple (r, g, b) de LED0 qui est rouge → (255, 0, 0)
- print(Adafruit_WS2801.RGB_to_color(coul[0],coul[1],coul[2])) # affiche le code couleur de LED0 → 255

Q2a

Bleu	Bleu	Bleu	Bleu	Bleu	Blanc	Blanc	Blanc	Blanc	Blanc	Rouge	Rouge	Rouge	Rouge	Rouge

Q2b

Vert	Jaune	Jaune	Vert	Jaune	Jaune	Vert	Jaune	Jaune	Vert	Jaune	Jaune	Vert	Jaune	Jaune

Q3a

```
def __init__(self, Pixel_COUNT : int):
    """ cré un bandeau de LEDs
    @param -- Pixel_COUNT, nombre de LEDs
    @return -- rien
    """
```

Q3b

Positionne LEDs6 et LED7 à Bleu