Proposition de correction

Exercice 1

Q1.a

La clé primaire de valeur 11 existe déjà

Q1.b

Mixe lettres + chiffres de taille 10 + 4 = 14

Q1.c

Lyon, 451 cours d'Emile Zola, 69100 Villeurbanne, 04 05 06 07 08

Q1.d

12

donne le nombre d'équipes

Q1.e

```
SELECT nom
FROM Equipe
ORDER BY nom
```

Q1.f

```
UPDATE Equipe
SET nom = "Tarbes"
WHERE id_equipe = 4
```

Q2.a

permet de mettre en relation la table Joueuse avec la clé primaire id_equipe de la table Equipe

Q2.b

Il faut faire une suppression en cascade car les clés étrangères de la table Joueuse pointe sur la table Equipe

Q2.c

```
SELECT Joueuse.nom, Joueuse.prenom
FROM Joueuse, Equipe
WHERE Equipe.nom = 'Angers'
AND Joueuse.id_equipe = Equipe.id_equipe
```

```
ORDER BY Joueuse.nom, Joueuse.prenom
```

Q3.a

Match

id_match	INT	
date	DATE	
#id_equipe1	INT	References Equipe(id_equipe)
score1	INT	
#id_equipe2	INT	References Equipe(id_equipe)
score2	INT	

Q3.b

```
INSERT INTO Match
VALUES(10, '2021-10-23', 3, 73, 6, 78)
```

Q4.a

Statistique

#id_joueuse	INT	References Joueuse(id_joueuse)
#id_match	INT	References Match(id_match)
point	INT	
rebond	INT	
passe	INT	

Q4.b

```
SELECT Equipe.nom, Joueuse.nom, Joueuse.prenom, Statistique.point, Statistique.rebond, Statistique.passe
FROM Equipe, Joueuse, Statistique
WHERE Equipe.id_equipe IN (SELECT DISTINCTROW id_equipe1, id_equipe2
        FROM Match
        WHERE id_match = 53)
AND Joueuse.id_equipe = Equipe.id_equipe
AND Statistique.id_joueuse = Joueuse.id_joueuse
ORDER BY  Equipe.nom
```

Exercice 2

Q1.a

11, 20, 32, 11, 20, 32, 11, 32, 11

Q1.b

11, 20, 32, 11, 32

Q2.a

liste_attente = [Processus(11, 4), Processus(20, 2), Processus(32, 3)]

Q2.b

```
def execute_un_cycle(self):
    """Met à jour le reste à faire après l'exécution d'un cycle."""
    self.reste_a_faire -= 1

def change_etat(self, nouvel_etat):
    """Change l'état du processus avec la valeur passée en paramètre."""
    self.etat =  nouvel_etat

def est_termine(self):
    """Renvoie True si le processus est terminé, False sinon, en se basant sur le reste à faire."""
    return self.reste_a_faire == 0
```

Q3.c

```
def tourniquet(liste_attente, quantum):
    ordre_execution = []
    while liste_attente != []:
        # On extrait le premier processus
        processus = liste_attente.pop(0)
        processus.change_etat("En cours d'exécution")
        compteur_tourniquet = 0
        while compteur_tourniquet < quantum and not processus.est_termine():
                ordre_execution.append(processus.pid)
                processus.execute_un_cycle()
                compteur_tourniquet = compteur_tourniquet + 1
        if not processus.est_termine():
                processus.change_etat("Suspendu")
                liste_attente.append(processus)
        else:
                processus.change_etat("Terminé")
    return ordre_execution
```

Exercice 3

Q1.a

importe la bibliothèque dans laquelle est définie la fonction sqrt()

Q1.b

Problème d'arrondi du nombre flottant.

0.1 ne peut s'exprimer en une suite finie en puissance de 2

Q1.c

le tuple est un objet immuable

Q2.a

```
from math import sqrt
class Segment:
    def __init__(self, point1, point2):
        self.p1 = point1
        self.p2 = point2
        self.longueur = sqrt((point2[0] – point1[0])**2 + ( point2[1] – point1[1])**2)
```

Q2.b

```
def liste_segments(liste_points):
    n = len(liste_points)
    segments = []
    for i in range(n):
        for j in range(i+1, n):
            # On construit le segment à partir des points i et j.
            seg = Segment(liste_points[i], liste_points[j])
            segments.append(seg)  # On l'ajoute à la liste
    return segments
```

Q2.c

1/2n(n-1)

id nombre d'arêtes dans un réseau

Q2.d

$O(n^2)$

Q3.a

```
def plus_court_segment(liste_segment):
    """renvoie l'objet Segment dont la longueur est la plus petite."""
    if len(liste_segment) == 1:
        return liste_segment[0]
    else:
        gauche = plus_court_segment(moitie_gauche(liste_segment))
        droite  = plus_court_segment(moitie_droite(liste_segment))
        return min(gauche.longueur, droite.longueur)
```

Q4.a

nuage_points = [(3,4), (2, 3), (−3, −1)]

Q4.b

segment = plus_court_segment(liste_segments(nuage_points))

print(f"({segment.p1[0]}, segment.p1[1])")

print(f"({segment.p2[0]}, segment.p2[1])")