

[Index des sujets 2023](#)

23-NSIJ1PO1 : Corrigé

Année : **2023**

Centre : **Polynésie**

Jour : **J1**

Enoncé : [!\[\]\(e3f8612927870f2e0f9f5989e6dd3064_img.jpg\) PDF](#)

1. Exercice 1 (4 points)

SQL

- La table contient déjà une entrée dont l'attribut `id_equipe` vaut 11. Comme il s'agit de la clé primaire cela provoque une erreur. C'est la contrainte d'unicité
 - L'attribut `telephone` est une chaîne de caractère limité à 20. On ne pouvait pas choisir des entiers car les numéros de téléphone commençant par 0 il disparaîtrait ainsi les espaces entre chaque paire de chiffres.
 - |Lyon | 451 cours d'Emile Zola,69100 Villeurbanne |04 05 06 07 08|
 - La requête le nombre d'entrée dans la table `equipe`
- Cette requête renvoie 12. Elle renvoie le nombre d'entités dans la relation `Equipe`.
-

Requête SQL

```
SELECT nom
FROM Equipe
ORDER BY noms;
```

f.

Requête SQL

```
UPDATE Equipe
SET nom='Tarbes'
WHERE id_equipe=4;
```

- L'attribut `id_equipe` a été déclaré clé étrangère de la relation `Joueuse` pour référence à la clé primaire `id_equipe` de la table `Equipe`.
- On ne peut pas supprimer directement l'équipe dans la relation `Equipe` car certaines entités de la relation `Joueuse` font référence à cette équipe : c'est la contrainte de référence.
-

Requête SQL

```
SELECT Joueuse.nom, Joueuse.prenom
FROM Joueuse
JOIN Equipe ON Equipe.id_equipe=Joueuse.id_equipe
WHERE Equipe.nom='Angers'
ORDER BY Joueuse.nom;
```

3. a. On peut proposer le schéma relationnel suivant : `Match (id_match : INT, date : DATE, #id_equipe_domicile : INT, #id_equipe_deplacement : INT, score_domicile : INT, score_deplacement : INT)`

#id_equipe_domicile et #id_equipe_deplacement sont des clés étrangères qui font référence à la relation Equipe.
b.

Requête SQL

```
INSERT INTO Match VALUES (10, "23/10/2021", 3, 6, 73, 78) ;
```

4. a. On peut proposer le schéma relationnel suivant : `Statistiques (id_stats : INT, #id_joueuse : INT, #id_match : INT, points : INT, passes_decisives : INT)`

b.

Requête SQL

```
SELECT Equipe.nom, Joueuse.nom, Joueuse.prenom, Statistiques.points, Statistiques.rebonds,
Statistiques.passes_decisives
FROM Statistiques
JOIN Joueuse ON Joueuse.id_joueuse = Statistiques.id_joueuse
JOIN Equipe ON Joueuse.id_equipe = Equipe.id_equipe
WHERE Statistiques.id_match = 53 ;
```

2. Exercice 2 (4 points)

Processus et POO

1. a. 11 - 20 - 32 - 11 - 20 - 32 - 11 - 32 - 11

b. 11 - 11 - 20 - 20 - 32 - 32 - 11 - 11 - 32

2. a.

Script Python

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

b.

Script Python

```

def execute_un_cycle(self):
    self.reste_a_faire-=1

def change_etat(self,nouvel_etat):
    self.etat=nouvel_etat

def est_termine(self):
    if self.reste_a_faire<=0:
        return True
    else:
        return False

```

C.

Script Python

```

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 processus.reste_a_faire > 0 and compteur_tourniquet < quantum :
            ordre_execution.append(processus.pid)
            processus.execute_un_cycle()
            compteur_tourniquet = compteur_tourniquet + 1
        if processus.reste_a_faire != 0:
            processus.change_etat("Suspendu")
            liste_attente.append(processus)
        else:
            processus.change_etat("Terminé")
    return ordre_execution

```

3. Exercice 3 (4 points)

POO et diviser pour régner

1. a. `from math import sqrt` permet d'import la fonction racine carrée de la bibliothèque math
- b. Cette instruction renvoie une False à cause des erreurs d'arrondis sur les nombres à virgule flottante. En effet, les nombres à virgule flottante sont représentés par une somme de puissance de 2. Comme 0,1, 0,2 et 0,3 ne peuvent pas s'exprimer comme une somme finie de puissance de 2, l'opération booléenne précédente renvoie False
- c. `point_A` est un tuple qui n'est pas mutable d'où l'erreur.

2. a.

Script Python

```

1  from math import sqrt
2  class Segment:
3      def __init__(self,point1,point2):
4          self.p1=point1
          self.p2=point2

```

```

5     self.longueur= sqrt((point1[0]-point2[0])**2+(point1[1]-point2[1])**2)
6

```

b.

Script Python

```

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

```

c. Pour liste de points de longueur n , on aura

$$(n-1) + (n-2) + \dots + 1 = (n-1) \times \frac{n-1+1}{2} = \frac{n \times (n-1)}{2} \text{ segments}$$

d. La complexité est donc de l'ordre de $O(n^2)$.

3. a.

Script Python

```

def plus_court_segment(liste_segments):
    if len(liste_segments)==1:
        return liste_segments[0]
    else:
        seg_gauche=plus_court_segment(moitie_gauche(liste_segments))
        seg_droite=plus_court_segment(moitie_droite(liste_segments))
    if seg_gauche.longueur>seg_droite.longueur:
        return seg_droite
    else:
        return seg_gauche

```

4. a.

Script Python

```

point_A=(3, 4)
point_B=(2, 3)
point_C=(-3, -1)

nuage_points=[point_A, point_B, point_C]

```

b.

Script Python

```

segment=plus_court_segment(liste_segments(nuage_points))

print((segment.p1[0],segment.p1[1]),(segment.p2[0],segment.p2[1]))

```

