b) Écrire la requête SQL qui permet l'insertion dans la table `Match` de l'enregistrement correspondant à l'exemple donné ci-dessus.

4. En plus du score final, sur la page web sont affichés des informations relatives aux performances des joueuses pendant le match.

Nous allons retenir ici seulement 3 critères : le nombre de points marqués, les rebonds et les passes décisives effectués.

Voici un extrait des statistiques du match n° 53 qui a opposé l'équipe de Landerneau à celle de Charleville-Mézières le 16/04/2022 :

Match n° 53
16/04/2022

Landerneau **56 | 64** **Charleville-Mézières**

Extrait statistiques :

Equipe	Nom	Prénom	Points	Rebonds	Passes décisives
Charleville-Mézières	Pouye	Tima	18	6	2
Charleville-Mézières	Akhator	Evelyn	15	17	0
Charleville-Mézières	Bouferra	Amel	10	3	9
Landerneau	Mane	Marie	18	2	3
Landerneau	Amukamara	Promise	12	2	5
Landerneau	Geiselsoder	Luisa	4	10	2

a) Proposer un schéma relationnel pour stocker les informations relatives aux statistiques des joueuses dans la base de données, telles que présentées ci-dessus.

b) Écrire la requête SQL qui a été utilisée pour afficher la partie « Extrait des statistiques »de l'exemple ci-dessus.

Exercice 2 — 4 points

Cet exercice porte sur la gestion des processus et la programmation orientée objet

On rappelle qu'un processus est l'instance d'un programme en cours d'exécution. Il est identifié par un numéro unique appelé PID. L'ordonnanceur est la composante du système d'exploitation qui gère l'allocation du processeur entre les différents processus. Nous allons nous intéresser à l'algorithme d'ordonnancement du tourniquet dont le fonctionnement est résumé ci-dessous :

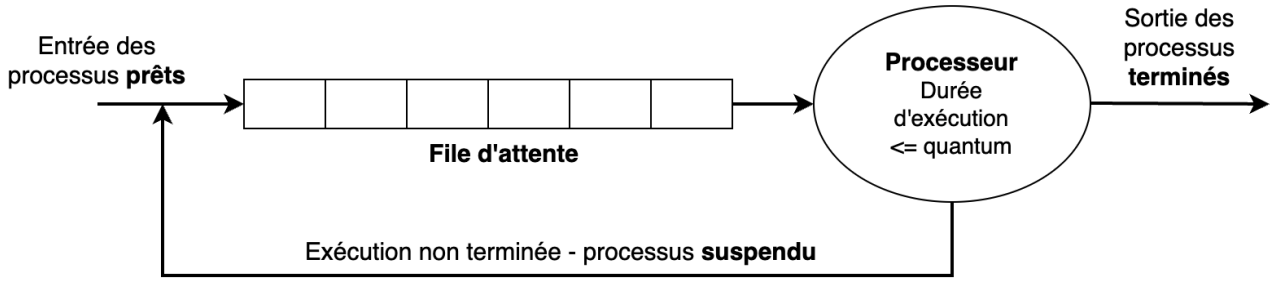

Schéma d'ordonnancement du tourniquet

- Les processus prêts à être exécutés sont placés dans une file d'attente selon leur ordre d'arrivée ;
- L'ordonnanceur alloue le processeur à chaque processus de la file d'attente un même nombre de cycles CPU, appelé **quantum** ;

- Si le processus n'est pas terminé au bout de ce temps, son exécution est suspendue et il est mis à la fin de la file d'attente ;
- Si le processus est terminé, il sort définitivement de la file d'attente.

1. On considère trois processus soumis à l'ordonnanceur **au même instant** pour lesquels on donne les informations ci-dessous :

PID	**Durée** (en cycles CPU)	**Ordre d'arrivée**
11	4	1
20	2	2
32	3	3

a) Si le quantum du tourniquet est d'un cycle CPU, recopier et compléter la suite des PID des processus dans l'ordre de leur exécution :
11, 20, 32, 11, ...

b) Donner la composition de la suite des PID lorsque le quantum du tourniquet est de deux cycles CPU.

2. L'objectif de la suite de l'exercice est d'implémenter en langage Python l'algorithme du tourniquet.
Nous allons utiliser une liste pour simuler la file d'attente des processus et la classe `Processus` dont le constructeur est donné ci-dessous :

```
class Processus:
    def __init__(self, pid, duree):
        self.pid = pid
        self.duree = duree
        # Le nombre de cycle qui restent à faire :
        self.reste_a_faire = duree
        self.etat = "Prêt"
```

Les états possibles d'un processus sont : « *Prêt* », « *En cours d'exécution* », « *Suspendu* »et « *Terminé* ».

a) Recopier et compléter l'instruction Python suivante permettant de créer la liste d'attente initiale des processus donnés dans le tableau précédent (le processus PID 11 est à l'indice 0 de la liste d'attente) :

```
liste_attente = [Processus(...,...), ..........., .........]
```

b) Recopier (sans les commentaires) et compléter les trois méthodes suivantes de la classe `Processus` :

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

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

    def est_termine(self):
```

```
        """Renvoie True si le processus est terminé, False sinon,
        en se basant sur le reste à faire."""
        ..............................................
```

c) La fonction `tourniquet` ci-dessous implémente l'algorithme décrit dans l'exercice.
Elle prend en paramètre une liste d'objets `Processus` donnés par ordre d'arrivée et un nombre entier positif correspondant au quantum. La fonction renvoie la liste des PID dans l'ordre de leur exécution par le processeur.
Recopier et compléter sur la copie le code manquant.

```
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 ...................... and ......................:
            ordre_execution.append(...........)
            processus.execute_un_cycle()
            compteur_tourniquet = compteur_tourniquet + 1
        if ......................:
            processus.change_etat("Suspendu")
            liste_attente.append(processus)
        else:
            processus.change_etat(...........)
    return ordre_execution
```

Exercice 3 — 4 points

Cet exercice porte sur l'algorithmique, la programmation orientée objet et la méthode diviser-pour-régner

L'objectif de cet exercice est de trouver les deux points les plus proches dans un nuage de points pour lesquels on connaît les coordonnées dans un repère orthogonal.

On rappelle que la distance entre deux points A et B de coordonnées $(x_A\,;\,y_A)$ et $(x_B\,;\,y_B)$ est donnée par la formule : $AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$.

Les coordonnées d'un point seront stockées dans un tuple de deux nombres réels.

Le nuage de points sera représenté en Python par une liste de tuples de taille n, n étant le nombre total de points. On suppose qu'il n'y a pas de points confondus (mêmes abscisses et mêmes ordonnées) et qu'il y a au moins deux points dans le nuage.

Pour calculer la racine carrée, on utilisera la fonction `sqrt` du module `math`, pour rappel :

```
>>> from math import sqrt
>>> sqrt(16)
```