

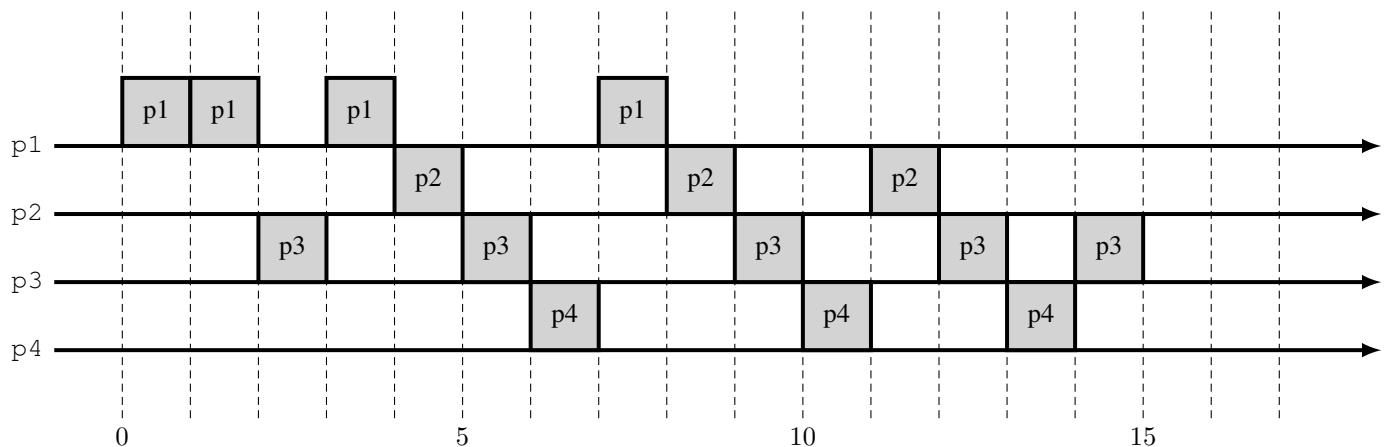
Amérique du nord - mai 2024 - sujet 1 (corrigé)

Exercice 1 (POO, files, processus)

1. Un processus peut être dans l'état prêt, élu ou bloqué.
2. Dans ce contexte précis, un processus ne peut être que prêt ou élu.
3. Le code suivant convient :

```
def defile(self):  
    if self.contenu == []:  
        return None  
    return self.contenu.pop(0)
```

4. On a le chronogramme suivant :



5. Le code suivant convient :

```
class Ordonnanceur:  
  
    def __init__(self):  
        self.temps = 0  
        self.file = File()  
  
    def ajoute_nouveau_processus(self, proc):  
        '''Ajoute un nouveau processus dans la file de  
        l'ordonnanceur'''  
        self.file.enqueue(proc)  
  
    def tourniquet(self):  
        '''Effectue une étape d'ordonnancement et renvoie le nom  
        du processus élu'''  
        self.temps += 1  
        if not self.file.est_vide():  
            proc = self.file.defile()  
            proc.execute_un_cycle()  
            if not proc.est_fini():  
                self.file.enqueue(proc)  
            return proc.nom  
        else:  
            return None
```

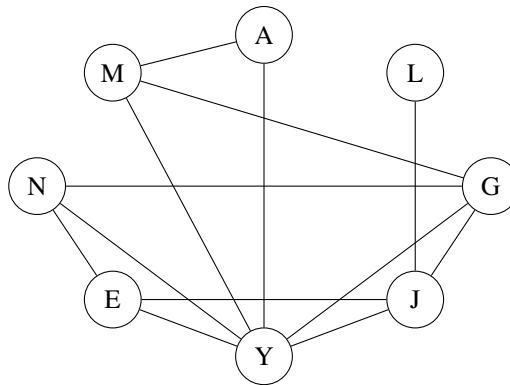
6. Le code suivant convient :

```
ordonnanceur = Ordonnanceur()
ordonnanceur.ajoute_nouveau_processus(depart_proc[0])
processus = ordonnanceur.tourniquet()
while processus != None:
    if ordonnanceur.temps in depart_proc:
        ordonnanceur.ajoute_nouveau_processus(depart_proc[ordonnanceur.temps])
    print(processus)
    processus = ordonnanceur.tourniquet()
```

7. Puisque le processus B et le processus D acquièrent respectivement le clavier et le fichier, le processus B est en attente du fichier à la ligne 2 (il est donc bloqué et ne peut libérer le clavier) et le processus D est en attente du clavier à la ligne 3 (il est donc bloqué et ne peut libérer le fichier). On a donc bien une situation d'interblocage.

Exercice 2 (Graphes)**Partie A : matrice d'adjacence.**

1. On a le graphe suivant :



2. On a la matrice d'adjacence suivante :

```
# sommets : G, J, Y, E, N, M, A, L
matrice_adj = [[0, 1, 1, 0, 1, 1, 0, 0], # G
               [1, 0, 1, 1, 0, 0, 0, 1], # J
               [1, 1, 0, 1, 1, 1, 1, 0], # Y
               [0, 1, 1, 0, 1, 0, 0, 0], # E
               [1, 0, 1, 1, 0, 0, 0, 0], # N
               [1, 0, 1, 0, 0, 0, 1, 0], # M
               [0, 0, 1, 0, 0, 1, 0, 0], # A
               [0, 1, 0, 0, 0, 0, 0, 0]] # L
```

3. La première instruction renvoie 1 car 'G' est à la première position dans la liste `sommets`. Quant à la seconde instruction, elle renvoie `None` car 'Z' n'est pas présent dans la liste `sommets`.

4. Le code suivant convient :

```
1 def nb_amis(L, m, s):
2     pos_s = position(L, s)
3     if pos_s == None:
4         return None
5     amis = 0
6     for i in range(len(m)):
7         amis += m[pos_s-1][i]
8     return amis
```

5. L'instruction renvoie 4.

Partie B : dictionnaire de listes d'adjacence.

6. Dans le dictionnaire $\{c: v\}$, c représente la clé et v la valeur associée.

7. On a le dictionnaire suivant :

```
graphe = {'G': ['J', 'Y', 'N', 'M'],
          'J': ['G', 'Y', 'E', 'L'],
          'Y': ['G', 'J', 'E', 'N', 'M', 'A'],
          'E': ['J', 'Y', 'N'],
          'N': ['G', 'Y', 'E'],
          'M': ['G', 'Y', 'A'],
          'A': ['Y', 'M'],
          'L': ['J']
          }
```

8. Le code suivant convient :

```
def nb_amis(d, s):  
    return len(d[s])
```

9. La liste du cercle d'amis de Lou est : Jade, Gabriel, Yanis, Emma, Nina.

10. Le code suivant convient :

```
1 def parcours_en_profondeur(d, s, visites = []):  
2     visites.append(s)  
3     for v in d[s]:  
4         if v not in visites  
5             parcours_en_profondeur(d, v, visites)  
6     return visites
```

Exercice 3 (Programmation Python, bases de données et SQL)**Partie A.**

1. Le séparateur choisi par l'étudiante est le point-virgule.
2. Le choix de ce séparateur est justifié par le fait qu'elle utilise déjà la virgule dans certaines réponses.
3. Le code suivant de la fonction `charger` convient :

```
def charger(nom_fichier):  
    with open(nom_fichier, 'r') as fichier:  
        donnees = list(csv.DictReader(fichier, delimiter = ';'))  
    return donnees
```

4. Il s'agit de la méthode `sleep` (lignes 37 et 39).
5. La variable `donnees[i]` est un dictionnaire.
6. Le code suivant convient :

```
flashcard = charger('flashcards.csv')  
d = choix_discipline(flashcard)  
c = choix_chapitre(flashcard, d)  
entrainement(flashcard, d, c)
```

Partie B.

7. La requête suivante convient :

```
INSERT INTO boite VALUES(5, 'tous les quinze jours', 15)
```

8. La requête suivante convient :

```
UPDATE flashcard SET reponse = '7 décembre 1941' WHERE id = 5
```

9. La requête suivante convient :

```
SELECT lib FROM discipline
```

10. La requête suivante convient :

```
SELECT chapitre.lib FROM chapitre  
JOIN discipline ON chapitre.id_disc = discipline.id  
WHERE discipline.lib = 'histoire'
```

11. La requête suivante convient :

```
SELECT flashcard.id FROM flashcard  
JOIN chapitre ON flashcard.id_ch = chapitre.id  
JOIN discipline ON chapitre.id_disc = discipline.id  
WHERE discipline.lib = 'histoire'
```

12. La requête suivante convient :

```
DELETE FROM flashcard WHERE id_boite = 3
```