# Proposition de correction

## Exercice 1

### Q1

Prêt (en attente), Endormi (bloqué), Élu (exécution)

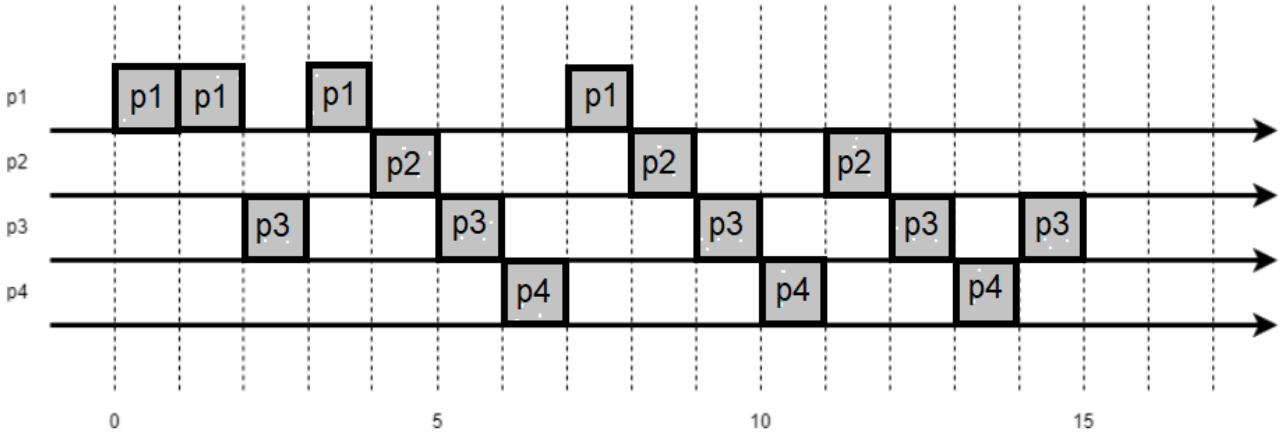
### Q2

Prêt, Élu

### Q3

```
def defile(self):
    return self.contenu.pop(0) if not self.est_vide() else None
```

### Q4




### Q5

```
class Ordonnanceur:
    def __init__(self):
        self.temps = 0
        self.file  = File()

    def ajoute_nouveau_processus(self, proc : Processus):
        """ Ajoute un nouveau processus dans la file de l'ordonnanceur. """
        self.file.enfile(proc)

    def tourniquet(self):
        """ Effectue une étape d'ordonnancement
        @return         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.enfile(proc)
            return proc.nom
        else:
            return None
```

## Q6

```
def lance_ordonnanceur(liste_proc : dict):
    timer     = 0
    scheduler = Ordonnanceur()
    scheduler.ajoute_nouveau_processus(depart_proc[timer])
    while scheduler.file.est_vide() is not True:
        processus = scheduler.tourniquet()
        print(processus, end=' ')
        timer += 1
        if timer in depart_proc:
            scheduler.ajoute_nouveau_processus(depart_proc[timer])
    print()
```

## Q7

D: acquérir le fichier

D: faire des calculs

B: acquérir le clavier

D: acquérir le clavier (en attente)

A: acquérir le GPU

B: acquérir le fichier (en attente)

D: libérer le clavier

C: acquérir le port

A: faire des calculs (en attente)

B: libérer le clavier

D: libérer le fichier

C: faire des calculs (en attente)

A: libérer le GPU

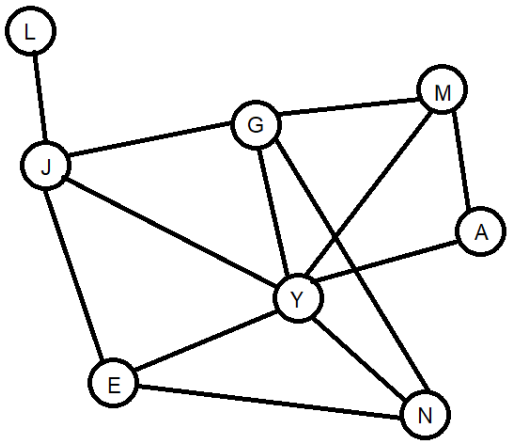
B: libérer le fichier

C: libérer le port

## Exercice 2

### Partie 1

#### *Q1*




#### *Q2*

```
matrice_adj = [
    #G, J, Y, E, N, M, A, L
    [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
```

#### *Q3*

0

None

#### *Q4*

```
def nb_amis(L : list, m : list, s : str) -> int:
    """
    @param L -- liste de noms de sommets
    @param m -- matrice d'adjacence d'un graphe
    @param s -- nom de sommet s
    @return le nombre d'amis du sommet s s'il est présent dans L et None sinon.
    """
    pos_s = position(L, s)
    if pos_s == None:
        return None
    amis = 0
    for i in range(len(m)):
        amis += m[i][pos_s]
```

```
    return amis
```

### Q5

4

## Partie 2

### Q6

- c : clé
- v : valeur

### Q7

```
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']
}
```

### Q8

```
def nb_amis(d : dict, s : str) -> int:
    """
    @param d -- dictionnaire d'adjacence
    @param s -- un nom de sommet
    @return le nombre d'amis du nom de sommet s
    @remark          On suppose que s est bien dans d.
    """
    return len(d[s])
```

### Q9

Emma, Gabriel, Jade, Nina et Yanis

### Q10

```
def parcours_en_profondeur(d, s, visites = []):
    """
    @param d -- dictionnaire d'adjacence
    @param s -- un sommet
    @return la liste des sommets issue du parcours en profondeur du graphe
    """
    visites += [s]
    for v in d[s]:
        if v not in visites:
            parcours_en_profondeur(d, v)
    return visites
```

## Exercice 3

### Partie 1

#### Q1

;

#### Q2

le caractère , se trouve dans plusieurs champ réponse

#### Q3

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

#### Q4

Sleep()

#### Q5

dict

#### Q6

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

### partie B

#### Q7

INSERT INTO boite

VALUES(5, ‘tous les quinze jours’, 15)

#### Q8

UPDATE flashcard

SET reponse = ‘Pearl Harbor – date, 7 décembre 1941’

WHERE reponse = ‘Pearl Harbor – date, 6 décembre 1941’

#### Q9

SELECT lib

FROM discipline

ORDER BY lib

#### Q10

SELECT chapitre.lib

FROM chapitre,  discipline

```
WHERE discipline.lib = 'histoire'
AND chapitre.id_disc = discipline.id
ORDER BY chapitre.lib
```

### *Q11*

```
SELECT flashcard.id
FROM flashcard, chapitre, discipline
WHERE discipline.lib = 'histoire'
AND chapitre.id_disc = discipline.id
AND flashcard.id_ch = chapitre.id
ORDER BY flashcard.id
```

### *Q12*

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