Exercice 2

Partie A

Q1

ps

Q2

PID

Q3

L'ordonnancement

Q4

kill

Partie B

Q1

P3	P3	P2	P1	P1	P1	P2	P2	P3	P3

Q2

Scénario 2

P1 ← R1
P3 ← R2
P1 ← R2
P3 ← R1 R1 jamais libéré par P1

Partie C

Q1.a

m = 0b 0110 0011 0100 0110 = 0x 63 46 = cF

Q1.b

m = 0b 0110 0011 0100 0110

k = 0b 1110 1110 1111 0000

c = 0b 1000 1101 1011 0110

Q2.a

a	b	a XOR b	(a XOR b) XOR b
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Exercice 4

Partie A

Q1

Programme 1 demande la table traçante et l'obtient.

Programme 2 demande le modem et l'obtient.

Programme 3 demande l'imprimante et l'obtient.

Programme 1 demande le modem et est en attente.

Programme 2 demande l'imprimante et est en attente.

Programme 3 demande la table traçante et est en attente.

→ tous les programmes s'attendent

Q2

demander (table traçante)
demander (imprimante)
exécution
libérer (imprimante)
libérer (table traçante)

Q3

bloqué

Partie B

Q1

ps –ef

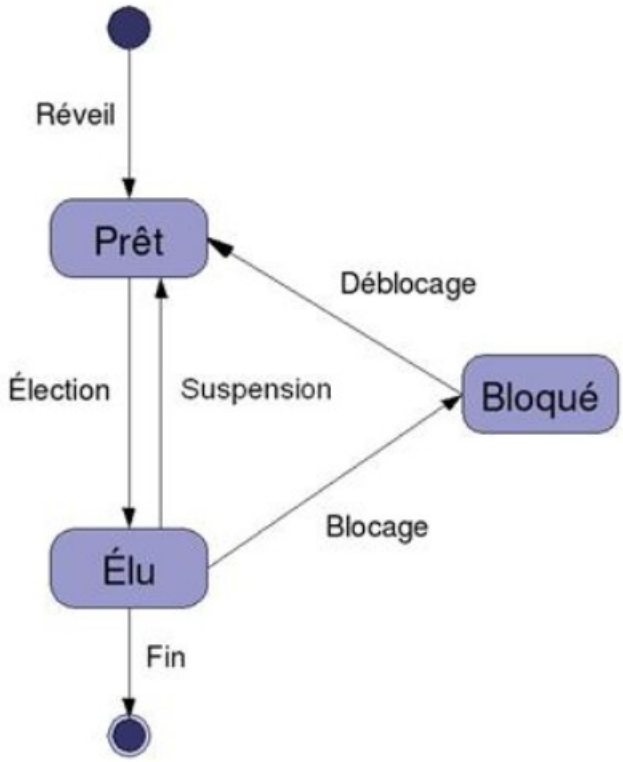
Q2

831

Q3

6211

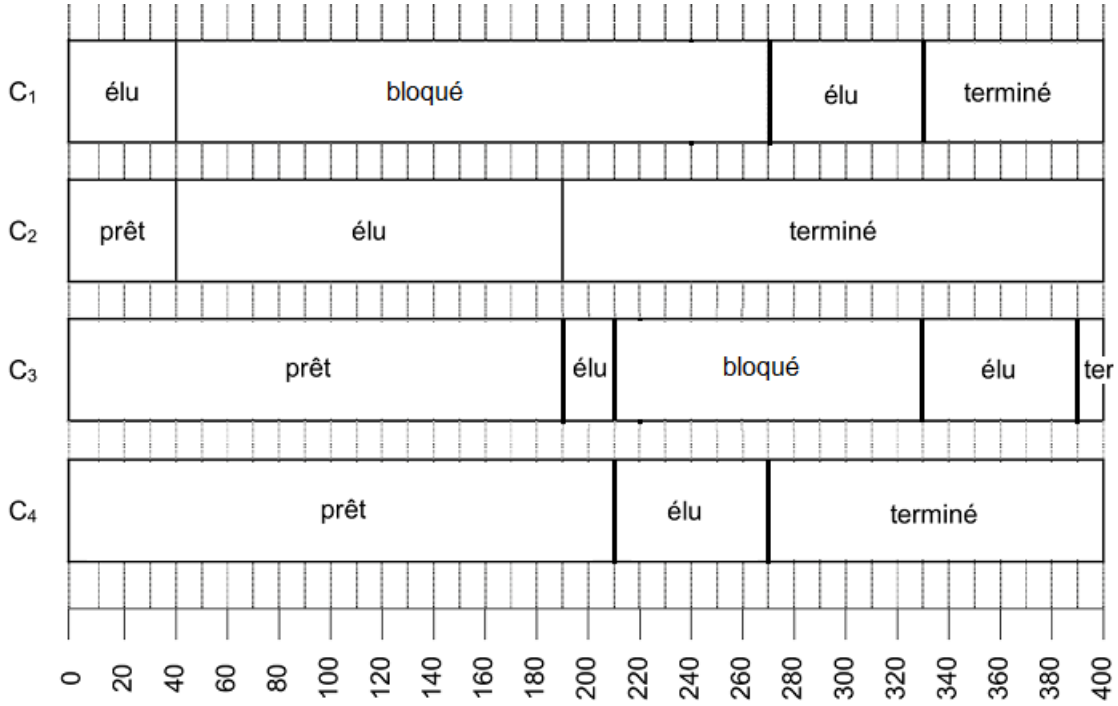
Q1b



Q2a

Premier entré, premier sorti (FIFO)

Q2b



```
        pile1.empiler(pile2.depiler())
```

Q4

```
def supprime_toutes_occurences(pile : object, element : int):
    """ supprime tous les éléments element de pile. """
    p = Pile()
    while not pile.est_vide():
        e = pile.depiler()
        if e != element:
            p.empiler(e)
    while not p.est_vide():
        pile.empiler(p.depiler())
```

Exercice 3

Partie A

Q1

init

Q2

- pid 5440 : python (Running)
- pid 5450 : bash (Running)

Q3

- bash (ppid 1912)
- python, bash, bash

Q4

- bash → python programme1.py
- bash → bash → python programme1.py

Q5

Non. python programme2.py est actif mais il peut devenir dormant (eg : attente de ressource)

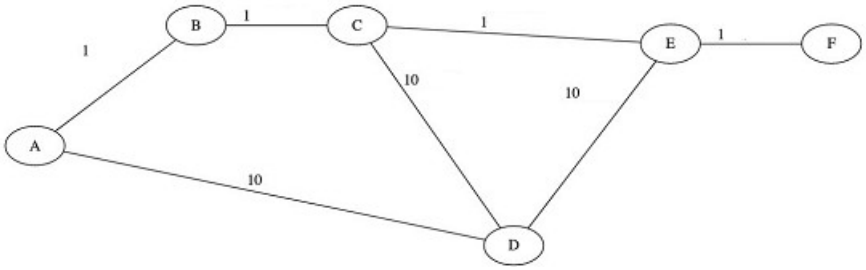
Partie B

Q1

Machine	Prochain saut	Distance
A	D	3
B	C	3
C	E	2
D	E	2

E	F	1

Q2



Machine	Prochain saut	Distance
A	B	4
B	C	2
C	E	2
D	E	11
E	F	1

Q3

OSPF car le chemin emprunté possède un débit de 100 Mb/s sur toute la ligne, alors que RIP emprunte une route qui limite le débit à 10 Mb/s.

Exercice 4

Partie A

Q1

lab2[1][0] = 2

Q2

```
def est_valide(i : int, j : int, n : int, m : int) -> bool:
    """ renvoie True si le couple (i, j) correspond à des coordonnées valides
    pour un labyrinthe de taille (n, m), et False sinon."""
    return -1 < i < n and -1 < j < m
```

Q3

```
def depart(lab : list) -> tuple:
    """ renvoie les coordonnées du départ d'un labyrinthe """
    n = len(lab)
    m = len(lab[0])
    for i in range(n):
        for j in range(m):
            if lab[i][j] == 2:
                return i,j
    return None
```