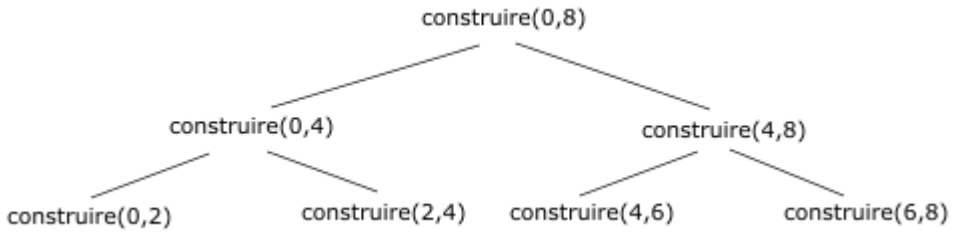
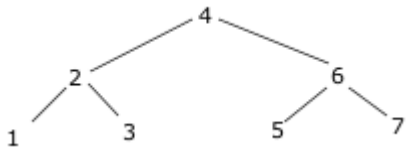
Éléments de correction sujet 06 (2023)

Exercice 1

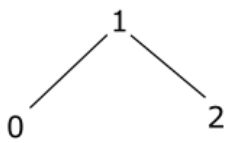
1.
   a. Chaque nœud a, au plus, 2 enfants, c'est donc bien un arbre binaire.
   b. Non, ce n'est pas un arbre binaire de recherche, car 4 est plus petit que 13 alors qu'il se trouve à sa droite.
2.
   a.
```
def construire(mini, maxi):
    assert isinstance(mini, int) and isinstance(maxi, int) and
mini <= maxi
    if maxi - mini == 1 or maxi - mini == 0:
        return Noeud(None, mini, None)
    elif maxi - mini == 2:
        return Noeud(None, (mini+maxi)//2, None)
    else:
        sag = construire(mini, (mini+maxi)//2)
        sad = construire((mini+maxi)//2, maxi)
        return Noeud(sag, (mini+maxi)//2, sad)
```
   b.



   c.



   d.



   e. Le parcours infixe d'un arbre binaire de recherche donne les nœuds dans un ordre croissant. On obtient ici : 1 - 2 - 3 - 4 - 5 - 6 - 7. Nous avons un ordre croissant, l'arbre du 2c est donc bien un arbre binaire de recherche.

f.

```
def maximum(abr):
    if abr is None:
        return None
    elif abr.droit is None:
        return maximum(abr.valeur)
    else :
        return maximum(abr.droit)
```

3.

a.

`mystere(abr_7_noeuds, 5, [])` renvoie [6, 4, 5]
`mystere(abr_7_noeuds, 6, [])` renvoie [6]
`mystere(abr_7_noeuds, 2, [])` renvoie `[]`

b.

La fonction *mystere* renvoie une liste contenant les nœuds à parcourir pour atteindre la valeur x. Si la valeur x n'est pas dans l'arbre, la fonction renvoie une liste vide.

Exercice 2

1.

a.

L'appareil à raclette à un id_objet de 4. Dans la table Possede on constate que l'id_objet 4 correspond à des id_membre de 1 et 2 qui correspondent (table Membre) à Mohamed Ali et Fernando Alonso.

b.

Pas d'id_membre égal à 5 dans Possede, donc Harry Kane ne propose aucun appareil.

2.

a.

'Dupont'	'Antoine'
'Kane'	'Harry'

b.

```
SELECT tarif
FROM Objet
WHERE description = "Scie circulaire"
```

c.

```
UPDATE Objet
SET tarif = 15
WHERE description = "Nettoyeur haute pression"
```

d.

```
INSERT INTO Membre
VALUES
(6, "Renard", "Wendie", "Villeurbanne", "69100")
```

3.

a.

On aura plusieurs fois le même couple (id_objet, id_membre) si une même personne loue plusieurs fois le même objet, ce qui n'est pas compatible avec la notion de clé primaire.

b.

l'id_membre 1 de la table Possede existe toujours malgré la suppression de

l'entrée Mohamed Ali de la table Membre. Cet id_membre 1 de la table Possede ne renvoie plus vers rien, ce qui va poser un problème.

c.

```
DELETE FROM Possede WHERE id_membre = 1
DELETE FROM Reservation WHERE id_membre = 1
```

4.

a.

```
SELECT COUNT(*)
FROM Reservation
JOIN Membre ON Reservation.id_membre = Membre.id_membre
WHERE nom = "Alonso" AND prenom = "Fernando"
```

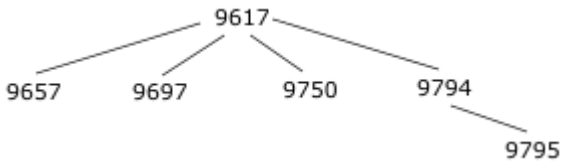
b.

```
SELECT nom, prenom
FROM Possede
JOIN Membre ON Membre.id_membre = Possede.id_membre
JOIN Objet ON Objet.id_objet = Possede.id_objet
WHERE description = "Appareil à raclette"
```

Exercice 3

1.

a.



b.

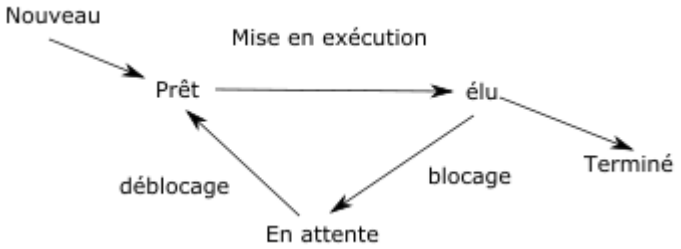
La commande est *bash*

c.

kill 9617 permet de détruire le processus 9617 et tous ses enfants.

2.

a.

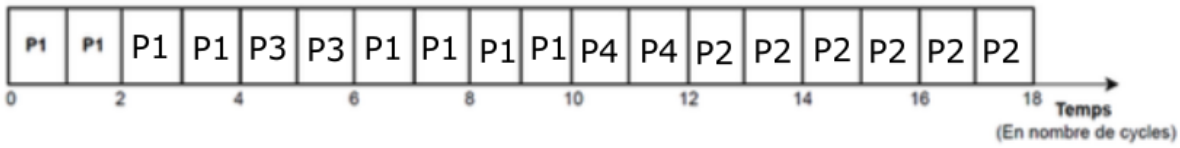


b.

Processus	temps d'exécution
P1	12 - 0 = 12
P2	18 - 2 = 16
P3	5 - 3 = 2
P4	9 - 7 = 2

moyenne des temps d'exécutions = (12+16+2+2) / 4 = 8

c.



d.

Processus	temps d'exécution
P1	10 - 0 = 10
P2	18 - 2 = 16
P3	6 - 3 = 3
P4	12 - 7 = 5

moyenne des temps d'exécutions = (10+16+3+5) / 4 = 8,5
La moyenne est supérieure, cet ordonnancement est donc moins performant.

3.

ATTENTION : les questions 3a et 3b ne sont pas claires du tout, je ne suis pas sûr et certain que les propositions ci-dessous sont celles qui étaient attendues.

a.

```
def choix_processus(liste_attente):
    if liste_attente != []:
        mini = len(liste_attente[0])
        indice = 0
        for i in range(1,len(liste_attente)):
            n = 0
            for e in liste_attente[i]:
                if e != '':
                    n = n + 1
            if n < mini :
                mini = n
                indice = i
    return indice
```

b.

```
def ordonnancement(liste_proc):
    execution = []
    attente = scrutation(liste_proc, [])
    while attente != []:
        indice = choix_processus(attente)
        if attente[indice] == []:
            del attente[indice]
        else:
            process_execute = attente[indice].pop()
            execution.append(process_execute)
        attente = scrutation(liste_proc, attente)
    return execution
```