

23-NSIJ2PO1 : Corrigé

Année : 2023

Centre : Polynésie

Jour : J2

Enoncé : 

1. Exercice 1 (4 points)

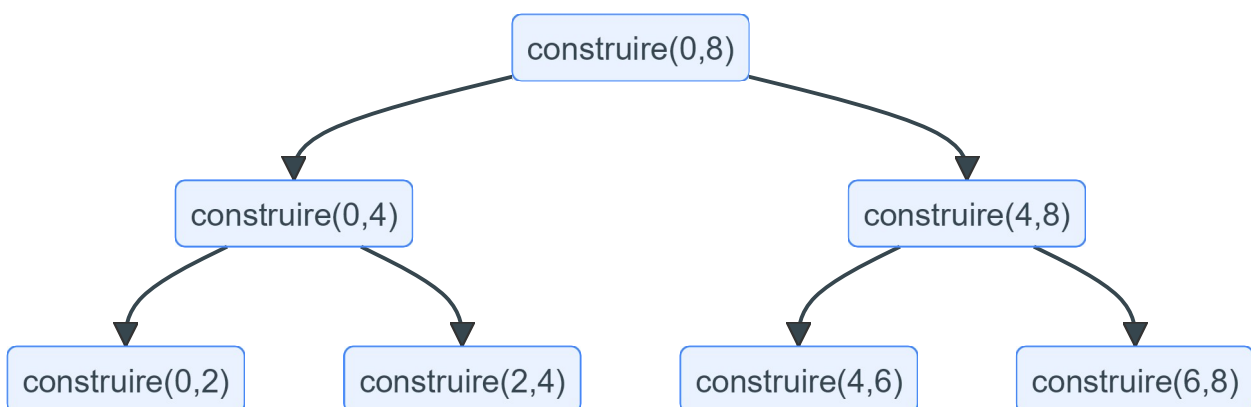
Arbres binaires, ABR, POO et récursivité

1. a. Cet arbre est un arbre binaire car chaque nœud possède au plus deux fils.
b. L'arbre ci-dessus n'est pas un arbre binaire de recherche car par exemple la valeur du nœud 4 est le fils droit de la racine dont la valeur est 13. Dans un arbre binaire de recherche chaque nœud du sous-arbre gauche a une valeur inférieure ou égale à celle du nœud considéré et chaque nœud du sous-arbre droit à une valeur supérieure ou égale à celle du nœud considéré.
2. a.

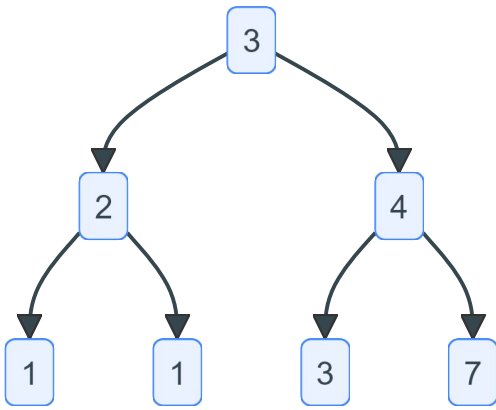
 Script Python

```
assert isinstance(mini, int) and isinstance(maxi, int) and mini <= maxi
```

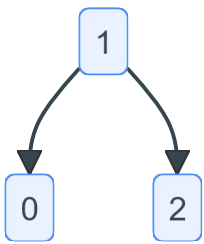
b.



c.



d.



e. Le parcours infixe de l'arbre binaire obtenu dans la question 2.c est :

1, 2, 3, 4, 5, 6, 7

C'est un arbre binaire de recherche car le parcours infixe parcourt les valeurs des nœuds dans l'ordre croissant

f.

```

Script Python

1 def maximum(abr):
2     if abr is None:
3         return None
4     elif abr.droit is None:
5         return abr.valeur
6     else:
7         return maximum(abr.droit)
  
```

2. a.

```

Script Python

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

b. La fonction `mystere` permet de déterminer le chemin vers la valeur `x` entrée en paramètre en partant de la racine de l'arbre si elle existe. Si la valeur `x` n'est pas dans l'arbre binaire de recherche, cette fonction renvoie une liste vide.

...

2. Exercice 2 (4 points)

Langage SQL

1. a. L'appareil à raclette a pour id 4. Dans la table Possede, les membres d'id 1 et 2, c'est-à-dire Ali Mohamed et Alonso Fernando, proposent à la location un appareil à raclette.
- b. Dans la table Possede, le membre d'id 5 n'est pas présent. Cela signifie qu'il ne propose pas d'objet à la location. Ce membre est Kane Harry.

2. a. Cette requête renvoie :

Dupont Antoine

Kane Harry

- b.

Requête SQL

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

- c.

Requête SQL

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

- d.

Requête SQL

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

3. a. Si ce couple était utilisé comme clé primaire, un membre ne pourrait réserver qu'une seule fois un même objet
- b. Ce membre est également présent dans les tables Possede et Reservation. Son attribut id_membre est clé étrangère de ces tables. Il faut donc d'abord supprimer ces entités des tables Possede et Reservation avant de pouvoir le supprimer de la table Membre.

- c.

Requête SQL

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

4. a.

Requête SQL

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

b.

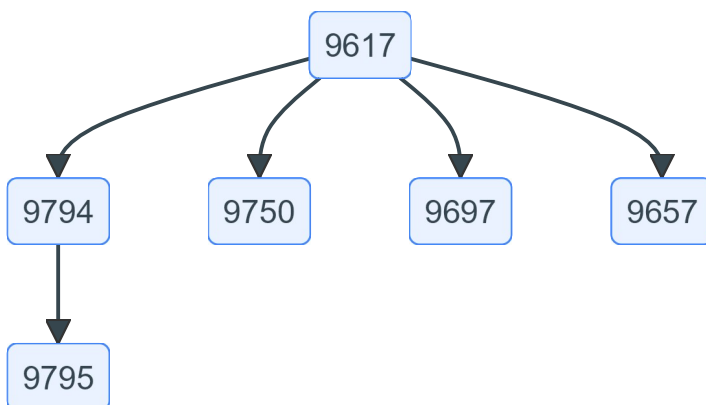
Requête SQL

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

3. Exercice 3 (4 points)

Programmation python

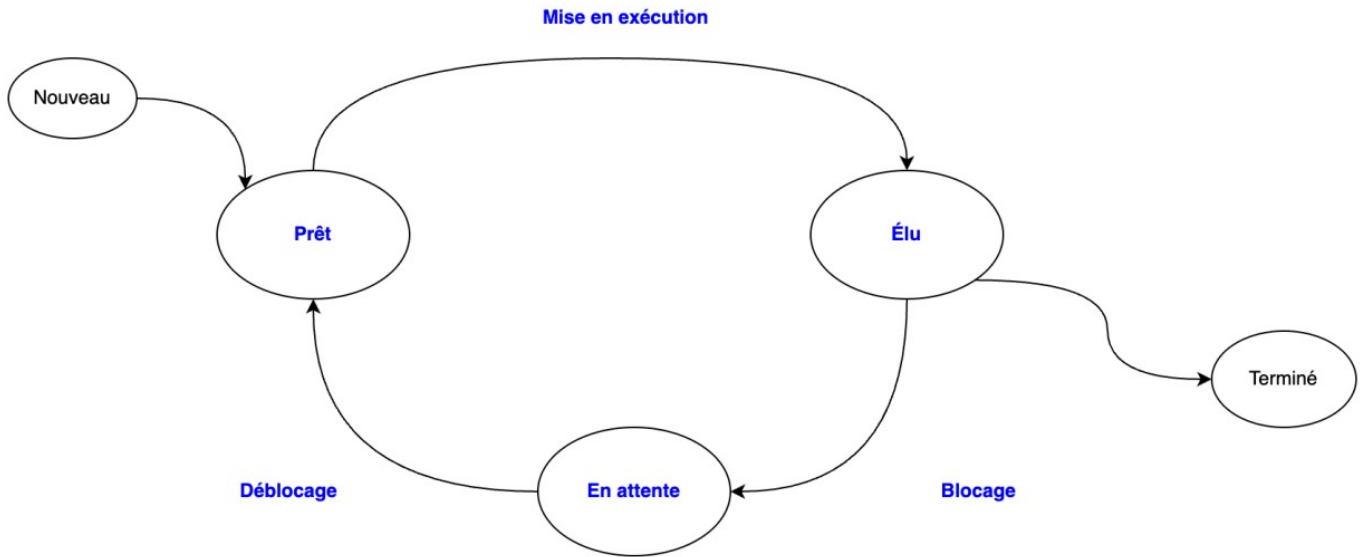
1. a.



b. La commande qui a lancé le premier processus de firefox est bash.

c. La commande permettant de supprimer tous les processus liés à firefox est kill 9617.

2. a.



b. Les temps d'exécution des quatre processus sont :

Processus	Instant d'arrivée	Instant de terminaison	Temps d'exécution
1	0	12	$12 - 0 = 12$
2	2	18	$18 - 2 = 16$
3	3	5	$5 - 3 = 2$
4	7	9	$9 - 7 = 2$

On obtient le temps d'exécution moyen : $\frac{12 + 16 + 2 + 2}{4} = 8$

c. P1-P1-P1-P1-P3-P3-P1-P1-P1-P1-P4-P4-P2-P2-P2-P2-P2-P2

d. Les temps d'exécution des quatre processus sont :

Processus	Instant d'arrivée	Instant de terminaison	Temps d'exécution
1	0	10	$10 - 0 = 10$
2	2	18	$18 - 2 = 16$
3	3	6	$6 - 3 = 3$
4	7	12	$10 - 7 = 5$

On obtient le temps d'exécution moyen : $\frac{10 + 16 + 3 + 5}{4} = 8.5$

Cet ordonnancement est moins performant que le précédent.

3. a.

```

Script Python
1 def choix_processus(liste_attente):
  
```

```

2 """Renvoie l'indice du processus le plus court parmi
3 ceux présents en liste d'attente liste_attente"""
4     if liste_attente != []:
5         mini = len(liste_attente[0])
6         indice = 0
7         # On parcourt les processus dans la liste d'attente
8         for i in range(1, len(liste_attente)):
9             # Si on trouve un processus plus court
10            if len(liste_attente[i]) < mini:
11                indice = i # On retient son indice
12                mini = len(liste_attente[i])
13    return indice

```

b.

Script Python

```

1 def ordonnancement(liste_proc):
2     """Exécute l'algorithme d'ordonnancement
3     liste_proc -- liste des processus
4     Renvoie la liste d'exécution des processus"""
5     execution = []
6     attente = scrutation(liste_proc, [])
7     while attente != []:
8         indice = choix_processus(attente)
9         # Retrait de la liste d'attente du dernier élément du
10        # processus le plus court
11        process_execute = attente[indice].pop()
12        # Le processus est entièrement fini, il est enlevé de la
13        # liste d'attente
14        if attente[indice] == []:
15            attente.pop(indice)
16            # On ajoute l'élément du processus choisi à la liste
17            # d'exécution
18            execution.append(process_execute)
19            attente = scrutation(liste_proc, attente)
20    return execution

```