Éléments de correction sujet 01 (2023)

Exercice 1

1. attributs de la table groupes : idgrp, nom, style, nb_pers

2. Le même nom peut apparaitre plusieurs fois (par exemple “Parker”), ce qui n’est pas possible avec une clé primaire

3. Cette requête renvoie : ‘Weather Report’ et ‘Return to Forever’

4.
```
UPDATE concerts
SET heure_fin = ‘22h30’
WHERE idconc = 36
```

5.
```
SELECT nom
FROM groupes
JOIN concerts ON concerts.idgrp = groupes.idgrp
WHERE scene = 1
```

6.
```
INSERT INTO groupes
VALUES
(15, 'Smooth Jazz Fourplay', ‘Free Jazz’, 4)
```

7.
```
def recherche_nom(tab):
    t = []
    for d in tab:
        if d['nb_concerts'] >= 4 :
            t.append(d['nom'])
    return t
```

Exercice 2

1. Nous avons un hôte d'adresse IP 172.16.2.3/24 qui appartient au réseau d'adresse réseau 172.16.2.0. C'est l'ordinateur d'Alice qui appartient à ce réseau (voir schéma).

2. cout = 10000/1000 = 10

3.

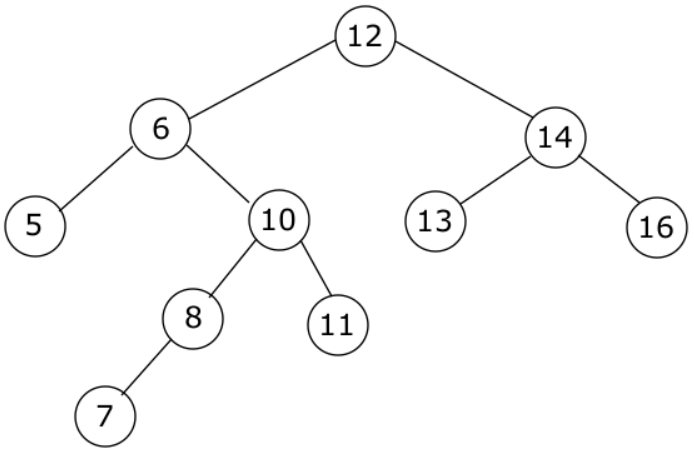
Routeur R6		
Destination	Pass	Cout
LAN1	R5	21
LAN2	-	-
WAN1	R5	11
WAN2	R5	20
WAN3	R5	11
WAN4	R5	12
WAN5	R5	10
WAN6	-	-
WAN7	-	-
WAN8	R5	10

4. Bob -> R1 -> R2 -> R5 -> R6 -> Alice

5. Le nouveau chemin est R1 -> R2 -> R4 - > R6. On évite le routeur R5, c'est donc le routeur 5 qui est en panne.

Exercice 3

1. Il s'agit d'une structure FIFO, c'est-à-dire une file
2.
   a. il s'agit de la taille d'un arbre
   b. il s'agit de la racine de l'arbre
   c. il s'agit de la feuille d'un arbre
3.
   a. attributs de la classe Noeud : tache, indice, gauche et droite
   b. La méthode *insere* est dite récursive, car elle s'appelle elle-même. Dans cette méthode récursive, on trouve bien le traitement du cas de base, ce qui permet d'affirmer que cette méthode se termine.
   c. il s'agit du signe > (strictement supérieur)
   d.



4.

```
def est_present(self, indice_recherche) :
      """renvoie True si l'indice de priorité indice_recherche
      (int) passé en paramètre est déjà l'indice d'un nœud
      de l'arbre, False sinon"""
      if self.est_vide():
          return False
      if self.racine.indice == indice_recherche:
          return True
      if self.racine.indice > indice_recherche :
          return self.racine.gauche.est_present(indice_recherche)
      else :
          return self.racine.droite.est_present(indice_recherche)
```

5.
   a.
   parcours infixe : 6 - 8 - 10 - 12 - 13 - 14
   b.
   Le parcours infixe permet d'obtenir les valeurs des nœuds d'un arbre binaire de recherche dans un ordre croissant. Le parcours infixe va donc permettre d'obtenir les tâches à accomplir dans l'ordre des priorités

6.

```
def tache_prioritaire(self):
        """renvoie la tache du noeud situé le plus
        à gauche de l'ABR supposé non vide"""
        if self.racine.gauche.est_vide(): #pas de nœud plus à
gauche
            return self.racine.tache
        else:
            return self.racine.gauche.tache_prioritaire()
```

7.