Éléments de correction sujet 14 (2022)

Exercice 1

Partie A

1. Arbre 1

Partie B

2.
   a. Sachant que les clés du sous arbre gauche sont inférieures ou égales à celle de la racine, le plus petit élément d'un ABR se situe le plus à gauche possible.
   b.
```
def RechercheValeur(cle, abr):
    if est_vide(abr):
        return False
    v = racine(abr)
    if v == cle:
        return True
    if v > cle :
        return RechercheValeur(cle, sous_arbre_gauche(abr))
    else :
        return RechercheValeur(cle, sous_arbre_droit(abr))
```
3.
   a.
   Il s'agit d'un parcours infixe
   b.
   parcours préfixe : 7 - 2 - 1 - 5 - 3 - 6 - 10 - 8 - 9
   c.
   parcours suffixe : 1 - 3 - 6 - 5 - 2 - 9 - 8 - 10 - 7
   d.
   parcours en largeur : 7 - 2 - 10 - 1 - 5 - 8 - 3 - 6 - 9

Exercice 2

Partie A

1.
   a. nous avons 5 *append*, donc 5 éléments dans la liste v
   b. La réponse attendue est “Les goélands”, mais cette instruction renvoie une erreur, car, *nom* est à la fois un attribut et une méthode, ce qui pose un problème. Il faudrait renommer la méthode, par exemple, en *getnom*.
   c.
```
def surface(self):
        return self.sejour.sup()+self.ch1.sup()+self.ch2.sup()
```
   ATTENTION : encore un problème de nommage de méthode : nous avons dans l’énoncé la méthode sup et l’attribut sup pour la classe Piece. Il faudrait renommer la méthode, par exemple, en *getsup*.

2.
```
def liste_cuis_equi(tab):
    for v in tab:
        if v.equip() == "eq":
            print(v.nom())
```
   ATTENTION : cette fonction ne fonctionne pas dans l’état à cause du problème évoqué à la question 1b

Partie B

3. appel d'une fonction par elle-même
4.
```
def max_surface(v):
    if len(v)<2:
        return v[0]
    else :
        if v[0].surface() < v[1].surface():
            del v[0]
        else :
            del v[1]
        return max_surface(v)
```

Exercice 3

Partie A

1. L'attribut num_Objet est uniqueb puisque qu'il est incrémenté d'une unité à chaque ajout d'un objet dans la BD. Il peut donc jouer le rôle de clé primaire.
2. Type (Type_Objet : String, Libelle_Objet : String)

Partie B

3. Seule la commande b ne provoque pas d'erreur (a -> problème à cause du '8' ; c -> problème à cause du WISEA J085510 ; d -> problème à cause du '133.781')
4. Ce code ne fonctionne pas, car l'attribut Type_objet possède déjà une entrée avec 'BD' (et Type_objet est une clé primaire).
5.

Proxima Cen b	768,067
Lalande 21185 b	392,753

6.
```
SELECT Nom_Objet, Libelle_Objet
FROM Gaia
JOIN Type ON Type.Type_Objet = Gaia.Type_Objet
WHERE Parallaxe > 400 AND  Libelle_Objet = 'Etoile'
```
7.
   a.
   ```
   INSERT INTO Type
   VALUES
   ('ST', 'Etoile')
   ```
   b. Voici les requêtes SQL à effectuer (dans cet ordre) :

   ```
   UPDATE Gaia
   SET Type_Objet = 'ST'
   WHERE Type_Objet = '*'
   ```

   puis

   ```
   DELETE FROM Type
   WHERE Type_Objet = '*'
   ```

Exercice 4

Partie A

1. schéma a

Partie B

2. PC02 appartient au réseau 192.168.10.0/24, l'adresse 192.168.10.2 peut donc convenir.
3. Seul l'octet de poids faible constitue la partie machine de l'adresse IP, il est donc possible de connecter 254 machines (256-2 = 254 ; on enlève 2 adresses : 192.168.10.0 (adresse réseau) et 192.168.10.255 (adresse de broadcast))
4. Un switch permet de connecter plusieurs machines à un même réseau local
5. Un routeur permet de relier plusieurs réseaux locaux entre eux.
6.

Destination Réseau	Passerelle	Métrique
192.168.10.0/24	0.0.0.0	0
2.100.40.0/24	2.100.40.1	1
3.100.30.0/24	3.100.30.2	1
4.10.10.0/24	4.10.10.2	1
4.20.10.0/24	3.100.30.2	2
7.30.40.0/24	3.100.30.2	3
6.10.30.0/24	2.100.40.1	2
90.10.20.0/24	2.100.40.1	2

7.

Destination Réseau	Passerelle	Métrique
90.10.20.0/24	4.10.10.2	4

Exercice 5

1. Principe FIFO donc Situation 2
2.
   a.
   V est une file [client3, client2, client1] (client1 correspond à la tête de file)
   F est une file [client4]
   val = 'Prioritaire'
   ATTENTION : il y a une erreur dans le programme proposé, il manque les 2 points au niveau du *while*
   b.

```
def longueur_file(F) :
    V= creer_file_vide()
    n=0
    while not est_vide (F) :
        n = n+1
        val = defiler(F)
        enfiler(V,val)
    while not est_vide(V) :
        val = defiler(V)
        enfiler(F, val)
    return n
```

   c.

```
def compter_prio(F) :
    V= creer_file_vide()
    n=0
    while not est_vide (F) :
        val = defiler(F)
        enfiler(V,val)
        if val == 'Prioritaire':
            n=n+1
    while not est_vide(V) :
        val = defiler(V)
        enfiler(F, val)
    return n
```