Métropole - septembre 2022 (corrigé)

Exercice 1 (Algorithmique et arbres binaires de recherche)
Partie A : préambule.

1. Il s'agit de l'arbre 1 (dans les deux autres arbres, les feuilles 4 et 3 du sous-arbre droit ne sont pas strictement supérieures à la valeur de la racine).

Partie B : analyse.

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) La fonction suivante convient :

```
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 (POO et récursivité)
Partie A : analyse du code et complétion.

1. (a) Il y a cinq `append`, donc cinq éléments dans la liste `v`.
   (b) On obtient le nom du deuxième élément de la liste `v`, donc `Les goélands`.
   (c) La fonction suivante convient :

```
def get_surface(self):
    return self.sejour.sup() + self.ch1.sup() + self.ch2.sup()
```

2. La fonction suivante convient :

```
def liste_cuis_equi(tab):
    for v in tab:
        if v.equip() == "eq":
            print(v.get_nom())
```

Partie B : récursivité.

3. Il s'agit de : « appel d'une fonction par elle-même ».
4. La fonction suivante convient :

```
def max_surface(v):
    if len(v)<2:
        return v[0]
    else :
        if v[0].get_surface() < v[1].get_surface():
            del(v[0])
        else :
            del(v[1])
        return max_surface(v)
```

Exercice 3 (Bases de données et SQL)
Partie A : schéma relationnel.

1. L'attribut `num_Objet` est unique, 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. On a le schéma relationnel suivant : `Type (Type_Objet : String, Libelle_Objet : String)`

Partie B : instructions SQL.

3. Seule l'instruction B ne provoque pas d'erreur. En effet :
   - ⋆ l'instruction A pose problème à cause du `'8'` écrit en chaîne de caractères ;
   - ⋆ l'instruction C pose problème à cause du `WISEA J085510` qui n'est pas écrit sous forme d'une chaîne de caractères ;
   - ⋆ l'instruction D pose problème à cause du `'133.781'` écrit en chaîne de caractères.
4. Ce code ne fonctionne pas, car l'attribut `Type_objet` est une clé primaire et possède déjà une entrée avec la valeur `'BD'`.
5. On obtient le résultat suivant :

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

6. La requête suivante convient :

```
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) La requête suivante convient :

```
INSERT INTO Type VALUES ('ST', 'Etoile')
```

   (b) Il faut saisir les requêtes suivantes dans cet ordre :

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

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

Exercice 4 (Processus et réseaux)
Partie A : architecture matérielle.

1. Il s'agit du schéma A.

Partie B : le réseau.

2. PC02 appartenant au réseau `192.168.10.0/24`, l'adresse `192.168.10.2/24` 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 ($256 - 2 = 254$ car on enlève deux 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. On a la table suivante :

Table de routage de R1		
Destination	Passerelle	Métrique
`4.20.10.0/24`	`3.100.30.2/24`	2
`7.30.40.0/24`	`3.100.30.2/24`	3
`6.10.30.0/24`	`2.100.40.1/24`	2

7. On a le tableau suivant :

Table de routage de R1		
Destination	Passerelle	Métrique
`90.10.20.0/24`	`4.10.10.2/24`	4

Exercice 5 (Files et programmation générale)

1. Il s'agit du principe FIFO et donc de la situation 2.
2. (a)
   - ⋆ `v` est une file `[Client3, Client2, Client1]` (`Client1` correspond à la tête de file) ;
   - ⋆ `F` est une file `[Client4]` ;
   - ⋆ `val` contient la valeur `Prioritaire`

   (b) Le code suivant convient :

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

   (c) Le code suivant convient :

```
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
```