Mayotte - mai 2022 - sujet 1 (corrigé)

Exercice 1 (Piles et files)

1. (a) On a la file `f` suivante :

1	4	3	8	2

(b) On a la pile `p` suivante :

5
8
6
2

(c) On a la pile `p` et la file `f` suivantes :

8	5	4	3	8	2	1

6
2

(d) On a la pile `p` et la file `f` suivantes :

4	3	8

2
1
5
8
6
2

2. A chaque tour de la première boucle, la file `f` perd un élément jusqu'à ce qu'elle soit vide. Puis, à chaque tour de la seconde boucle, la file `f` gagne un élément. A l'issue de la fonction, le contenu de la file `f` est

4	3	2	1

.
La pile renvoyée par la fonction `mystere` est vide.

3. (a) On a le tableau suivant :

f	2 | 1 | 3	2 | 1	3 | 2	3	2 | 3	1 | 2 | 3
p	⌴	3	1	2 1	1	⌴

(b) Cet algorithme permet de trier dans l'ordre décroissant (plus grand élément en tête de file) une file composée de trois éléments (pour quatre éléments ou plus cela ne fonctionne pas ! ?). Si on prend une file contentant au départ plus de trois éléments, on peut dire que cet algorithme permet de mélanger cette file.

Exercice 2 (POO)

1. (a) La fonction suivante convient :

```
def donnePremierIndiceLibre(Mousse):
    """
    Mousse est une liste.
    La fonction doit renvoyer lindice du premier
    emplacement libre (contenant None) dans la liste Mousse
    ou renvoyer 6 en labsence dun emplacement libre dans
    Mousse.
    """
    i = 0
    while ..............  and Mousse[i] != None :
        ..............
    return i
```

(b) La fonction suivante convient :

```
def placeBulle(B):
    i = donnePremierIndiceLibre(Mousse)
    if i != len(Mousse):
        Mousse[i] = B
```

2. La fonction suivante convient :

```
def bullesEnContact(B1, B2):
    return distanceEntreBullles(B1, B2) <= (B1.rayon + B2.rayon)
```

3. La fonction suivante convient :

```
SELECT NumChambre
FROM Reservations
WHERE date(DateArr) <= date('2020-12-28') AND date(DateDep) > date('2020-12-28')
```

Exercice 3 (Bases de données et SQL)

1. (a) La requête renvoie les titres des QCM dont la date de création est postérieure au 10 janvier 2022. Elle renvoie donc `POO` et `Arbre parcours`.

   (b) La requête suivante convient :

```
SELECT note FROM lien_eleve_qcm WHERE ideleve = 4
```

2. (a) Si par exemple, l'élève d'`ideleve` 4 fait deux fois le QCM d'`idqcm` 3 nous aurons deux fois le couple `(4,3)` dans la table `lien_eleve_qcm`, ce qui pose problème, car il n'est pas possible d'avoir deux fois le même couple `(ideleve, idqcm)` (clé primaire).

   (b) On doit rajouter la ligne : `4, 2, 18` à la table `lien_eleve_qcm`.

   (c) La requête suivante convient :

```
INSERT INTO eleves VALUES (6, "Lefèvre", "Kevin")
```

   (d) La requête suivante convient :

```
DELETE FROM lien_eleve_qcm WHERE ideleve = 2
```

3. (a) La requête suivante convient :

```
SELECT ............................. FROM eleves
JOIN lien_eleve_qcm ON eleves.ideleve = ..............................
WHERE ............................. ;
```

   (b) La requête affiche :

```
Dubois Thomas
Marty Mael
Bikila Abebe
```

4. La requête suivante convient :

```
SELECT nom, prenom, note
FROM lien_eleve_qcm
JOIN eleves ON eleves.ideleve = lien_eleve_qcm.ideleve
JOIN qcm ON qcm.idqcm = lien_eleve_qcm.idqcm
WHERE titre = 'Arbre parcours'
```

Exercice 4 (Arbres binaires)

1. (a) Chaque personne a deux parents (qui peuvent être connus ou inconnus) qui ont eux-mêmes deux parents, etc. On retrouve donc bien la structure d'un arbre binaire où un nœud a, au plus, deux enfants.

   (b) Dans un arbre binaire de recherche, on retrouve une notion d'ordre des nœuds que l'on ne retrouve pas dans un arbre généalogique.

2. (a) Parcours préfixe : Albert Normand - Jules Normand - Michel Normand - Jules Normand - Odile Picard - Hélène Breton - Evariste Breton

   (b) Parcours infixe : Jules Normand - Michel Normand - Odile Picard - Jules Normand - Evariste Breton - Hélène Breton - Camélia Charentais

   (c) La fonction suivante convient :

```
def parcours(racine_de_l_arbre) :
    if racine_de_l_arbre != None :
        noeud_actuel = racine_de_l_arbre
        print(noeud_actuel.identite[0]+" "+noeud_actuel.identite[1])
        parcours(noeud_actuel.gauche)
        parcours(noeud_actuel.droite)
```

   (d) La fonction suivante convient :

```
def parcours(racine_de_l_arbre) :
    if racine_de_l_arbre != None :
        noeud_actuel = racine_de_l_arbre
        parcours(noeud_actuel.gauche)
        print(noeud_actuel.identite[0]+" "+noeud_actuel.identite[1])
        parcours(noeud_actuel.droite)
```

3. (a) La fonction suivante convient :

```
class Noeud() :
    def __init__(prenom, nom) :
        self.identite = (prenom, nom)
        self.gauche = None
        self.droite = None
        ..............................
```

   (b) La fonction suivante convient :

```
fonction mystereABR(a, n)
    si a est null alors
        renvoyer Faux
    sinon si billet(a) vaut n alors
        renvoyer Vrai
    sinon si billet(a) strictement inférieur à n alors
        renvoyer mystereABR(filsgauche(a))
    sinon
        renvoyer mystereABR(filsdroit(a))
    fin si
fin fonction
```

4. L'ordre d'affichage est : Odile, Hélène, Camélia, Marie, Eulalie, Gabrielle, Janet

Exercice 5 (Réseau et protocoles de routage)

1. (a) Il faut 4 octets pour constituer une adresse IPv4.

   (b) Comme on est en `/24`, le masque de sous-réseau est `255.255.255.0`.

2. On a le tableau suivant :

PC3	172								150								4								30							
	1	0	1	0	1	1	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0
Mas.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Pour obtenir l'adresse réseau binaire, on réalise un `ET` logique entre chaque bit de l'adresse IP (ligne 2) et du masque de sous-réseau (ligne 3)																																
Rés.	1	0	1	0	1	1	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	172								150								4								0							

3. (a) Les adresses IP possibles sont `172.150.4.11` et `172.150.4.200`. En effet, la 1 n'appartient pas au même réseau, la 2 est déjà utilisée, la 3 n'existe pas (on ne peut pas coder 257 avec un octet) et la 5 est l'adresse du réseau (donc non utilisable par une machine).

   (b) Il faut saisir la commande `ipconfig` sous Windows ou `ifconfig` sous Unix.

4. Il serait nécessaire d'entièrement reconfigurer toutes les machines du réseau 1 ou du réseau 2 pour que toutes les machines aient la même adresse réseau. Mais il serait beaucoup plus simple d'utiliser un routeur pour relier les deux réseaux.

5. La fonction suivante convient :

```
>>> liste_IP=[[192,168,10,1],[192,168,10,25],[192,168,10,13]]
>>> adresse([192,168,10,3],liste_IP)
pas trouvée, ajoutée
>>> liste_IP
[[192,168,10,1],[192,168,10,25],[192,168,10,13],[192,168,10,3]]
>>> adresse([192,168,10,25],liste_IP)
trouvée
>>> liste_IP
[[192,168,10,1],[192,168,10,25],[192,168,10,13],[192,168,10,3]]
```