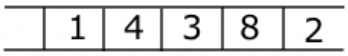
Éléments de correction sujet 10 (2022)

Exercice 1

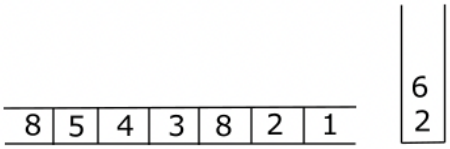
1.

a.

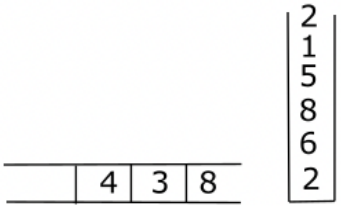


b.

c.

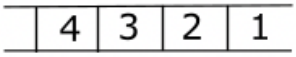


d.



2.

contenu de la file f :



La pile renvoyée par la fonction mystère est vide.

3.

a.

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 3 éléments (pour 4 éléments ou plus cela ne fonctionne pas !?). Si on prend une file contentant au départ plus de 3 éléments, on peut dire que cet algorithme permet de mélanger cette file.

Exercice 2

1. a.

```
def donnePremierIndiceLibre(Mousse):
    i=0
    while i < len(Mousse) and Mousse[i] != None :
          i = i + 1
    return i
```

b.

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

2.

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

3.

```
def collision(indPetite, indGrosse, mousse) :
    surfPetite = pi*Mousse [indPetite].rayon**2
    surfGrosse = pi*Mousse [indGrosse].rayon**2
    surfGrosseApresCollision = surfPetite + surfGrosse
    rayonGrosseApresCollision = sqrt(surfGrosseApresCollision/pi)
    Mousse[indGrosse].dirx = Mousse[indGrosse].dirx / 2
    Mousse [indGrosse].diry = Mousse [indGrosse].diry / 2
    Mousse[indPetite] = None
```

Exercice 3

1.
   a. POO et Arbre Parcours
   b.

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

2.
   a. Si par exemple, l'élève d'ideleve 4 fait 2 fois le QCM d'idqcm 3 nous aurons 2 fois le couple (4,3) dans la table lien_eleve_qcm, ce qui pose problème, car il n'est pas possible d'avoir 2 fois le même couple (ideleve - idqcm) (clé primaire).
   b. On doit rajouter la ligne : 4, 2, 18 à la table lien_eleve_qcm
   c.

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

   d.

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

3.
   a.

```
SELECT nom, prenom
FROM eleves
JOIN lien_eleve_qcm ON eleves.ideleve = lien_eleve_qcm.ideleve
WHERE idqcm = 4
```

   b.

Dubois	Thomas
Marty	Mael
Bikila	Abebe

4.

```
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

1.
   a. Chaque personne a 2 parents (qui peuvent être connus ou inconnus) qui ont eux-mêmes 2 parents… 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.

```
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.

```
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. il manque le self dans les arguments de la méthode __init__

```
class Noeud() :
    def __init__(prenom, nom) :
        self.identite = (prenom, nom)
        self.gauche = None
        self.droite = None
        self.generation = 0
```

   b.

```
def numerotation(racine_de_l_arbre, num_gen=0) :
    noeud_actuel = racine_de_l_arbre
    if noeud_actuel != None :
        noeud_actuel.generation = num_gen
        numerotation(noeud_actuel.gauche, num_gen+1)
        numerotation(noeud_actuel.droite, num_gen+1)
```

4. Ordre d'affichage : Odile, Hélène, Camélia, Marie, Eulalie, Gabrielle, Janet

Exercice 5

1.
   a. Il faut 4 octets pour constituer une adresse IPv4
   b. 255.255.255.0

2.

Adresse IP (V4) du PC3	Ligne 1	172								150								4								30					
	Ligne 2	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
Masque de sous réseau	Ligne 3	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 (ligne3)																															
Adresse du réseau	Ligne 4	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
	Ligne 5	172								150								4								0					

3.
   a.
      liste des adresses IP possibles : 172.150.4.11 et 172.150.4.200. 1) n'appartient au même réseau ; 2) est déjà utilisée ; 3) avec 1 octet, il n'est pas possible de coder 257 ; 5) c'est l'adresse du réseau (non utilisable par une machine)
   b.
      ipconfig sous Windows et ifconfig sous Unix

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

5.

```
def adresse(addr, liste):
    if addr in liste :
        print("trouvée")
    else :
       print("pas trouvée, ajoutée")
       liste.append(addr)
```