BAC NSI – Mayotte 1 – 2022

Exercice 1 : PILE et FILE

Question 1 :

La file suivante est appelée `f` :

4	3	8	2	1

La pile suivante est appelée `p` :

5
8
6
2

a- `enfiler(f, defiler(f))`

La file devient :

1	4	3	8	2

b- `empiler(p, depiler(p))`

La pile ne change pas :

5
8
6
2

c-
```
for i in range(2):
    enfiler(f, depiler(p))
```

La file et la pile deviennent :

8	5	4	3	8	2	1

6
2

d-
```
for i in range(2):
    empiler(p, defiler(f))
```

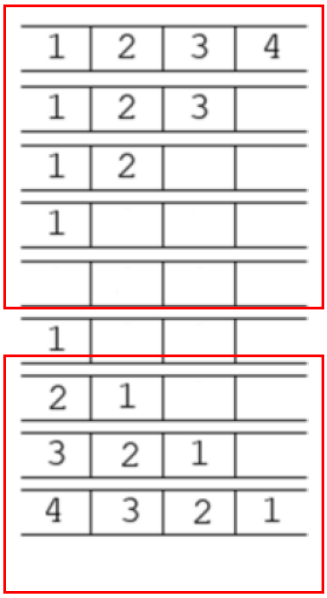
La file et la pile deviennent :

4	3	8

2
1
5
8
6
2

Question 2 : L'état de la file f au cours de l'exécution de la fonction mystère est :

```
def mystere(f):
    p = creer_pile_vide()
    while not est_file_vide(f):
        empiler(p, defiler(f))
    while not est_pile_vide(p):
        enfiler(f, depiler(p))
    return p
```



La pile p retournée par cette fonction est vide.

Question 3 : L'exécution de la fonction knuth() pour la file 2 , 1 , 3 donne :

a- N = 3

		$i = 0$	$i = 1$			$i = 2$			
File f	2 , 1 , 3	2 , 1	2	3 , 2	3 , 2	3	3	2 , 3	1 , 2 , 3
Pile p		|3|	|3|		|1|	|1|	|2| |1|	|1|	
e			1	1	1	2	2		

b- Cet algorithme tri les valeurs de la file par ordre croissant.

Exercice 2 : Bases de données

Question 1 :

a- La requête affiche les titres des QCM créés après le 10 janvier 2022 soit :

- POO
- Arbre Parcours

b- Une requête qui donne les notes de l'élève qui a pour identifiant 4 est :

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

Question 2 :

a- Chaque clé primaire doit être **unique**. Ainsi il ne peut y avoir qu'un seul couple (ideleve , idqcm)

Un élève ne peut donc pas faire deux fois le même qcm.

b- L'élève Marty Mael est déjà enregistré dans la table eleves. Le qcm sur la POO est aussi déjà enregistré dans la table qcm. Il reste ainsi à créer un nouvel enregistrement dans la table lien_eleve_qcm avec une requête INSERT.

c- Pour mettre à jour la base de données, il suffit de créer l'enregistrement correspondant à ce nouvel arrivant dans la table eleves :

```
INSERT INTO eleves  VALUES (6 , 'Lefèvre' ,  'Kévin')
```

On peut aussi écrire :

```
INSERT INTO eleves(ideleves , nom , prenom)  VALUES (6 , 'Lefèvre' ,  'Kévin')
```

d- Pour supprimer les références à cet élève dans la table lien_eleve_qcm, on exécute la requête suivante :

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

Rq : Cette dernière requête n'était pas exigée dans l'énoncé de la question.

Question 3 :

a- La requête suivante affiche la liste des noms et prénoms des élèves ayant fait le QCM d'idqcm égal à 4

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

b- Cette requête renvoie :

- Dubois Thomas
- Marty Mael
- Bikila Abebe

Question 4 : Une requête qui affiche le nom, le prénom et la note des élèves ayant fait le QCM Arbre Binaire est :

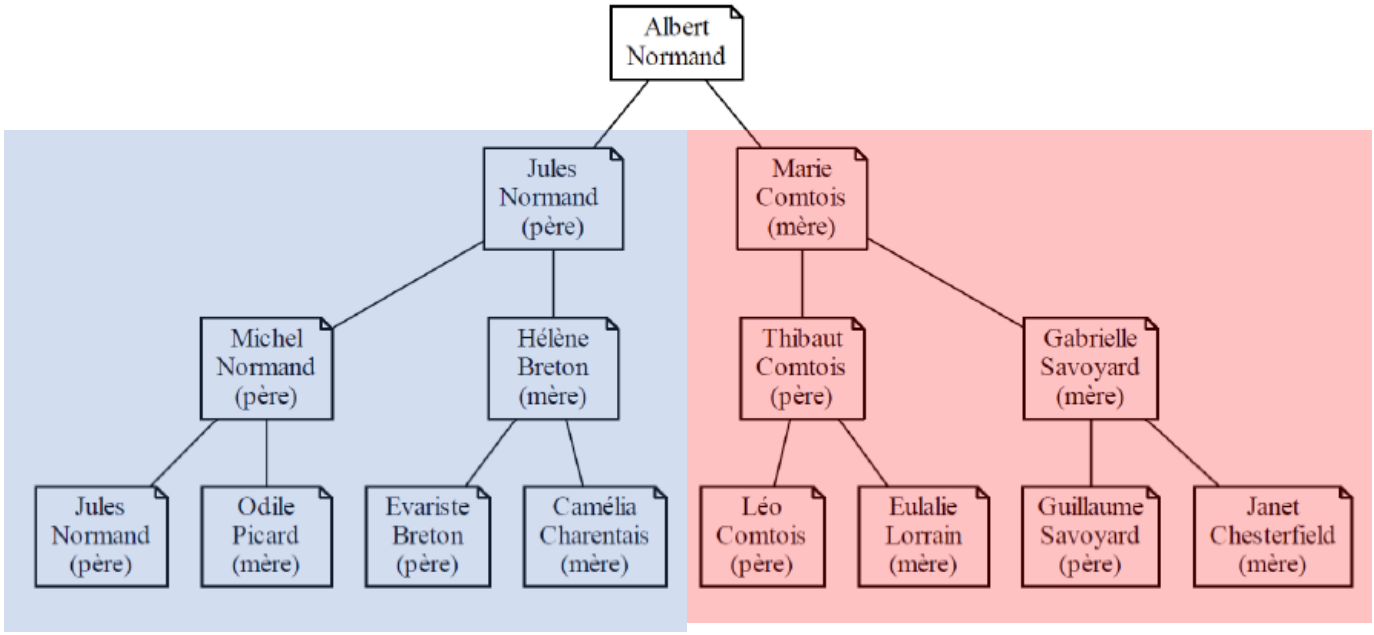
```
SELECT eleves .nom , eleves .prenom , lien_eleve_qcm.note   FROM   eleves
JOIN  lien_eleve_qcm  ON  eleves.ideleve  =  lien_eleve_qcm.ideleve
JOIN  qcm  ON  lien_eleve_qcm. idqcm  = qcm .idqcm
WHERE qcm.titre = 'Arbre Binaire'
```

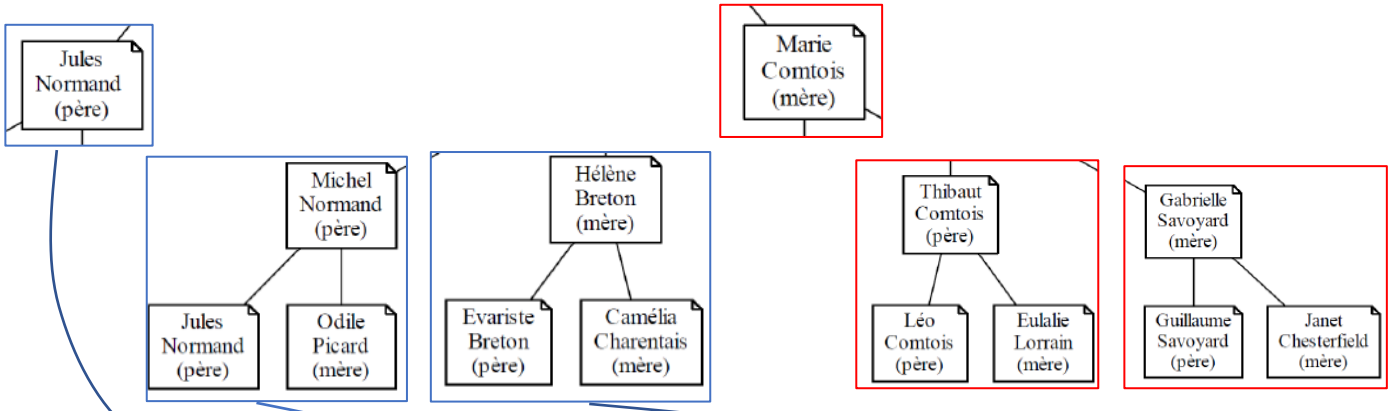
Exercice 3 : Arbres binaires

Question 1 : Cet arbre généalogique est un arbre binaire car chaque nœud a au plus 2 enfants.

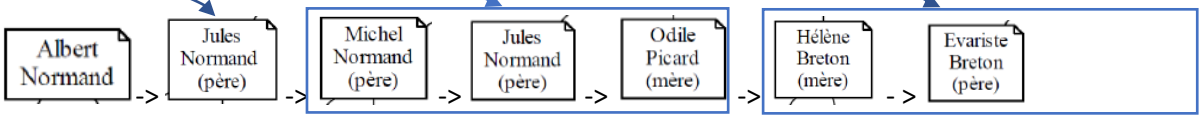
Question 2 :

a- Le parcours en profondeur préfixe est un parcours récursif qui analyse un arbre binaire dans l'ordre : Nœud -> Fils gauche -> Fils droit

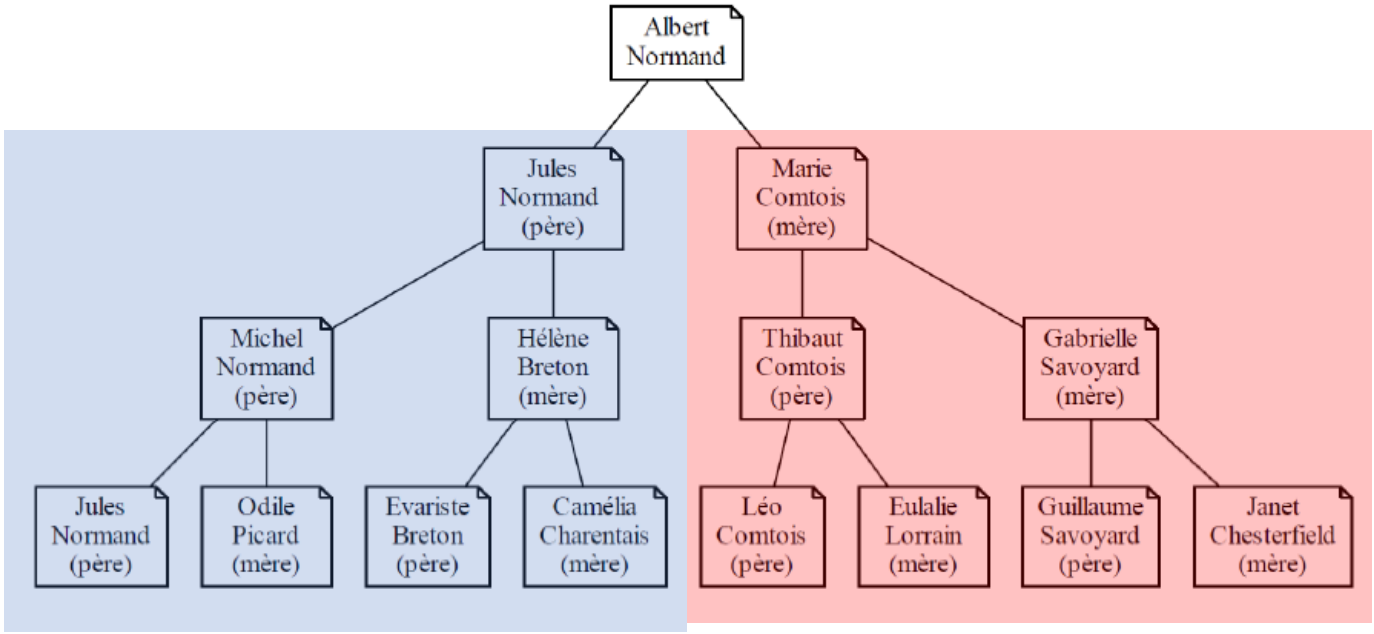




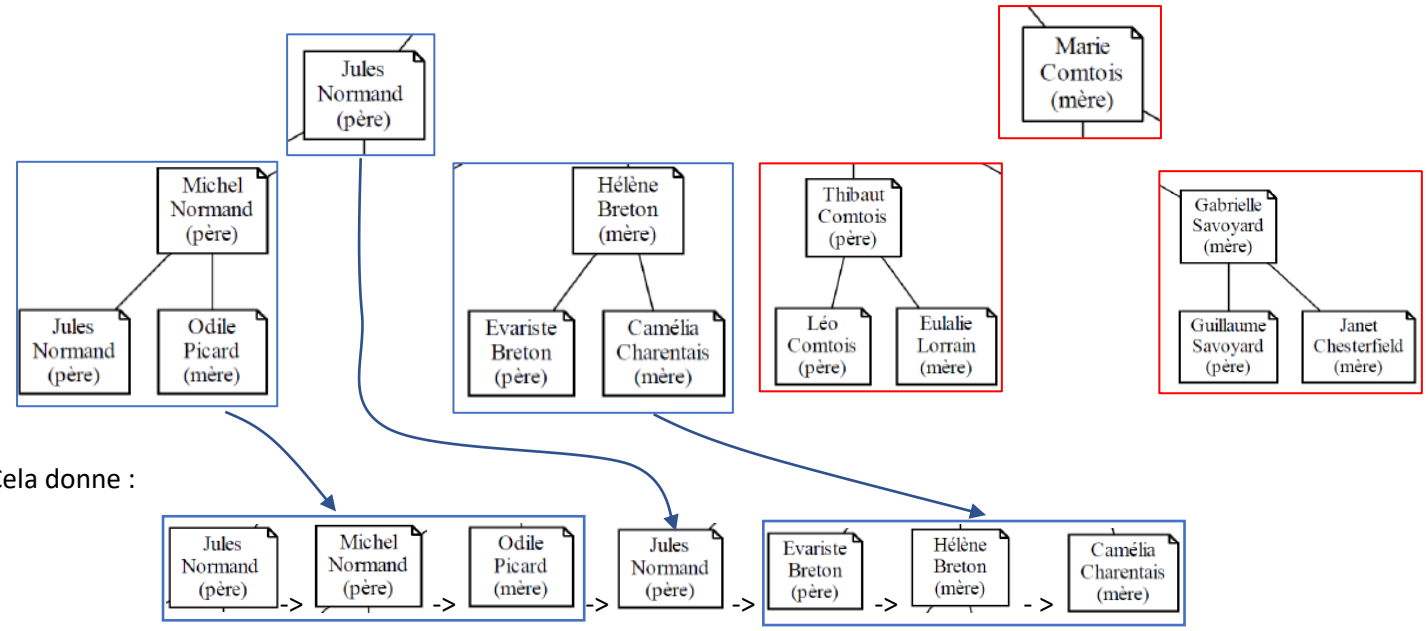
Cela donne :



b- Le parcours en profondeur infixe est un parcours récursif qui analyse un arbre binaire dans l'ordre : Fils gauche -> Nœud -> Fils droit



Cela donne :



c- Pour un parcours préfixe, on obtient la fonction suivante :

```
def parcours(racine_de_l_arbre) :
    if racine_de_l_arbre != None :
        noeud_actuel = racine_de_l_arbre
        print(noeud_actuel.identite)
        parcours(noeud_actuel.gauche)
        parcours(noeud_actuel.droit)
```

d- Pour un parcours infixe, on obtient la fonction suivante :

```
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)
        parcours(noeud_actuel.droit)
```

Question 3 :

a- On ajoute un attribut génération :

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

b- La fonction récursive numérotation() peut être la suivante :

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

Les appels récursifs de la fonction numerotation() s'enchainent de la façon suivante ::

numerotation(Albert Normand,0)

AlbertNormand.generation = 0

numerotation(Jules Normand,1)
JulesNormand.generation = 1

numerotation(Michel Normand,2)
JulesNormand.generation = 2

numerotation(Jules Normand,3)
JulesNormand.generation = 3

numerotation(Odile Picard,3)
OdilePicard.generation = 3

numerotation(Hélène Breton,2)
HélèneBreton generation = 2

numerotation(Evariste Breton,3)
EvaristeBreton generation = 3

numerotation(Camélia Charentais,3)
CaméliaChatentais generation = 3

numerotation(Marie Comtois,1)
MarieComtois.generation = 1

numerotation(Thibault Comtois,2)
ThibaultComtois. generation = 2

numerotation(Léo Comtois,3)
LéoComtois. generation = 3

numerotation(Eulalie Lorrain,3)
EulalieLorrain. generation = 3

numerotation(Gabrielle Savoyard,2)
GabrielleSavoyard. generation = 2

numerotation(Guillaume Savoyard,3)
GuillaumeSavoyard. generation = 3

numerotation(Janet Chesterfield,3)
JanetChesterfield. generation = 3

Question 4 : On a la fonction suivante

```
def mystere(N,affiche) :
    if N != None :
        if affiche :
            print( N.identite[0])
        mystere(N.gauche,False)
        mystere(N.droite,True)
```

En exécutant la ligne `mystere(racine_de_l_arbre,False)`
Les appels récursifs de la fonction mystere() s'enchainent de la façon suivante :

mystere(Albert Normand,False)

numerotation(Jules Normand,**False**)

numerotation(Michel Normand,**False**)

numerotation(Jules Normand,**False**)

numerotation(Odile Picard,**True**)
print(OdilePicard.identite[0])

numerotation(Hélène Breton,**True**)
print(HélèneBreton.identite[0])

numerotation(Evariste Breton,**False**)

numerotation(Camélia Charentais,**True**)
print(CaméliaChatentais.identite[0])

numerotation(Marie Comtois,**True**)
print(MarieComtois.identite[0])

numerotation(Thibault Comtois,**False**)

numerotation(Léo Comtois,**False**)

numerotation(Eulalie Lorrain,**True**)
print(EulalieLorrain.identite[0])

numerotation(Gabrielle Savoyard,**True**)
print(GabrielleSavoyard.identite[0])

numerotation(Guillaume Savoyard,**False**)

numerotation(Janet Chesterfield,**True**)
print(JanetChesterfield.identite[0])

L'affichage dans la console est ainsi le suivant :

```
>>> (executing file "bacEssai.py")
Odile
Hélène
Camélia
Marie
Eulalie
Gabrielle
Janet
```