Éléments de correction sujet 08 (2023)

Exercice 1

1.

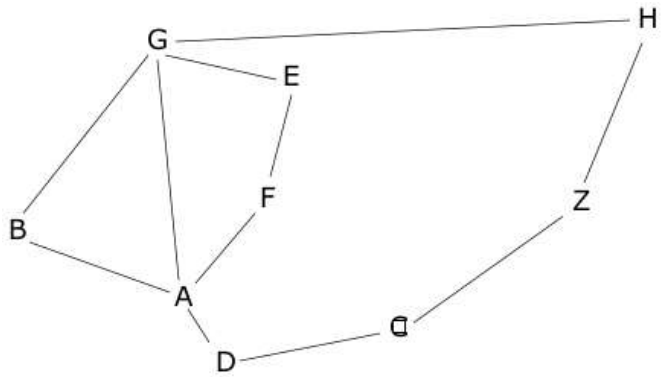
Noeuds B	
Destination	Coût
A	1
C	3
D	2
E	2
F	2
G	1
H	2

Noeuds F	
Destination	Coût
A	1
B	2
C	3
D	2
E	1
G	2
H	3

2.

F - E - G - H
F - A - G - H

3.



4.

On recherche le chemin qui a le plus faible coût :
B - G - E - F - A - D - C - H avec un coût de 34 (par exemple le chemin B - G - H ne sera pas retenu, car il a un coût de 101, même chose pour le chemin B - A - D - C - H qui a un coût de 40)

Exercice 2

1.

a. Une clé primaire est un attribut dont la valeur permet d'identifier de manière unique un t-uplet de la relation

b.

La valeur 3 a déjà été utilisé pour l'attribut *id_astronaute* de la table *Astronaute*. Nous allons donc avoir une erreur puisque *id_astronaute* est la clé primaire de la table Astronaute

c.

Fusee (id_fusee : int, modele : TEXT, constructeur : TEXT, nb_places : INT)

2.

a.

Cette requête renvoie 2

b.

```
SELECT modele, constructeur
FROM Fusee
WHERE nb_places > 3
```

c.

```
SELECT nom, prenom
FROM Astronaute
ORDER BY nom
```

3.

a.

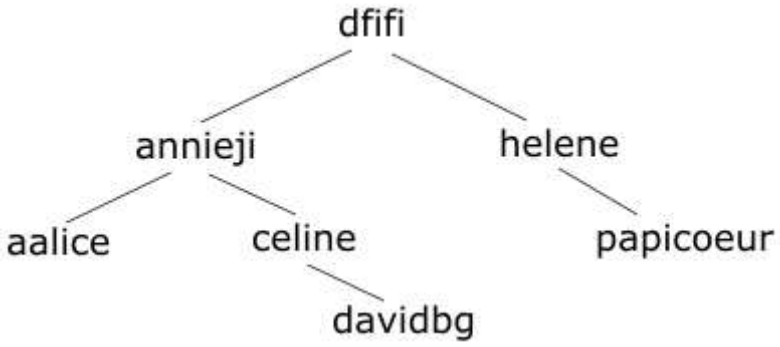
INSERT INTO Vol VALUES(5, 3, “12/04/2023”);
INSERT INTO Equipe VALUES(5, 1);
INSERT INTO Equipe VALUES(5, 4);

b.

```
SELECT nom, prenom
FROM Equipe
JOIN Vol ON Vol.id_vol = Equipe.id_vol
JOIN Astronaute ON Astronaute.id_astronaute =
Equipe.id_astronaute
WHERE Date = “25/10/2022”
```

Exercice 3

Partie 1

1. taille de l'arbre = 5 ; hauteur de l'arbre = 3
2. 

3. C - Parcours en profondeur dans l'ordre infixe
4.

```
def present(self, identifiant):
    if self.est_vide():
        return False
    elif self.racine() == identifiant:
        return True
    elif self.racine() < identifiant:
        return self.sd().present(identifiant)
    else:
        return self.sg().present(identifiant)
```

Partie 2

5.
   a. `est_vide(f1)` renvoie False
   b. après la commande defiler(f1), on obtient la file f1 suivante :

	'bac'	'nsi'	'2023'

   c. f2 correspond à la file suivante :
   'poule' | 'python' | 'castor'
6.

```
def longueur(f):
    resultat = 0
    g = creer_file()
    while not(est_vide(f)) :
        elt = defiler(f)
        resultat = resultat + 1
        enfiler(g , elt)
    while not(est_vide(g)):
        enfiler(f, defiler(g))
    return resultat
```

7. C - '2!@59fgds'
8.
```
def ajouter_mot(f, mdp):
    enfiler(f,mdp)
    if longueur(f) > 3:
        defiler(f)
```
9.
```
def mot_file(f, mdp):
    g = creer_file()
    present = False
    while not(est_vide(f)):
        elt = defiler(f)
        enfiler(g, elt)
        if elt == mdp:
            present = True
    while not(est_vide(g)):
        enfiler(f, defiler(g))
    return present
```
10.
```
def modification(f, nv_mdp):
    if not mot_file(f, nv_mdp) and est_valide(nv_mdp):
        ajouter_mot(f, nv_mdp)
        return True
    else :
        return False
```