Éléments de correction sujet 08

Exercice 1

1a
Cette requête permet d'obtenir la liste des salles et la liste des marques :

012	HP
114	Lenovo
223	Dell
223	Dell
223	Dell

1b

Gen-24	012
Tech-62	114
Gen-132	223

2

```
SELECT *
FROM Ordinateur
WHERE annee >= 2017 ORDER BY annee
```

3a
L'attribut salle ne peut pas être une clé primaire, car on retrouve dans la table Ordinateur plusieurs fois la même salle (par exemple la salle 223).

3b
Imprimante (nom_imprimante : String, marque_imp : String, modele_imp : String, salle : String, #nom_ordi : String)

Dans la table Ordinateur les ordinateurs ont un nom unique (clé primaire), il est donc possible d'utiliser Imprimante.nom_ordi comme clé étrangère

4a

```
INSERT INTO Videoprojecteur
(salle, marque_video, modele_video, tni)
VALUES
('315', 'NEC', 'ME402X', false);
```

4b

```
SELECT Ordinateur.salle, Ordinateur.nom_ordi,
Videoprojecteur.marque_video
FROM Ordinateur
INNER JOIN Videoprojecteur ON Videoprojecteur.salle = Ordinateur.salle
WHERE Ordinateur.video = true AND Videoprojecteur.tni = true
```

Exercice 2

1

- faible consommation d'énergie par rapport à une architecture classique
- miniaturisation par rapport à une architecture classique
- plus faible coût de fabrication qu'une architecture classique

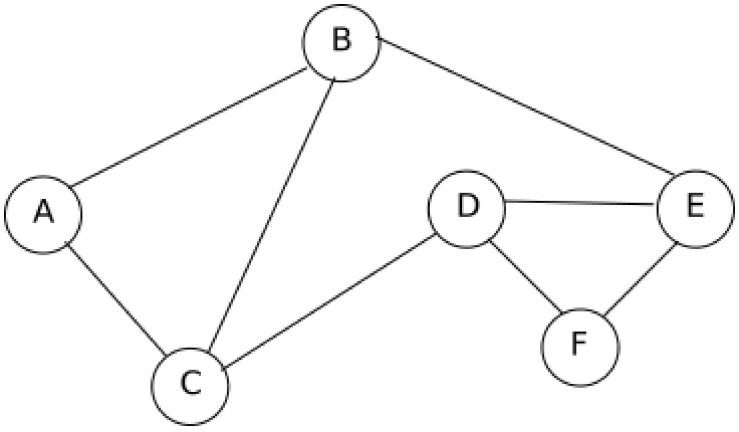
2

La donnée D2 est utilisée par le SGBD. Dans le même temps, le traitement de texte est en attente de cette même donnée D2. Le SGBD ne peut pas libérer la donnée D2 car il est en attente de la donnée D3 et D4. Cette même donnée D4 est utilisée par le logiciel de CAO qui est lui-même en attente de la donnée D5. Cette donnée D5 est utilisée par le tableur qui est en attente de la donnée D1 alors que cette donnée D1 est utilisée par le tableur. Le tableur est en attente de la donnée D1, donnée D1 utilisée par le traitement de texte. Nous avons donc bien les applications qui s'attendent mutuellement.
Si un programme A utilise une donnée D et est en attente d'une autre donnée D' alors que dans le même temps un programme B utilise la donnée D' et est en attente de la donnée D, on parle alors d'une situation d'interblocage. C'est cette situation d'interblocage qui nous est décrite dans cette question.

3

A -> B -> E -> F

4



Exercice 3

1a

```
def total_hors_reduction(tab):
    total = 0
    for pa in tab:
        total = total +  pa
    return total
```

on acceptera aussi un simple :

```
def total_hors_reduction(tab):
    return sum(tab)
```

1b

```
def offre_bienvenue(tab):
    somme = 0
    longueur=len(tab)
    if longueur > 0:
        somme = tab[0]*0.8
    if longueur > 1:
        somme = somme + tab[1]*0.7
    if longueur > 2:
        for i in range(2,longueur):
            somme=somme+tab[i]
    return somme
```

2

```
def prix_solde(tab):
    longueur = len(tab)
    reduc = 1
    if longueur == 1 :
        reduc = 0.9
    if longueur == 2 :
        reduc = 0.8
    if longueur == 3 :
        reduc = 0.7
    if longueur == 4 :
        reduc = 0.6
    if longueur >= 5 :
        reduc = 0.5
    return reduc*total_hors_reduction(tab)
```

autre possibilité plus “courte” :

```
def prix_soldeb(tab):
    return total_hors_reduction(tab)*(1-0.1*min(5,len(tab)))
```

3a
Dans cette question nous partons du principe que tab n'est pas vide.

```
def minimum(tab):
    mini = tab[0]
    for pa in tab:
        if pa <  mini:
            mini = pa
    return mini
```

on acceptera aussi :

```
def minimum(tab):
    return min(tab)
```

3b

```
def offre_bon_client(tab):
    longueur = len(tab)
    total = total_hors_reduction(tab)
    if longueur >= 2:
        total = total - minimum(tab)
    return total
```

4a

plusieurs solutions possibles
[30.5, 20.0, 35.0, 15.0, 6.0, 5.0, 10.5] => total après promotion = 122 - 20 - 5 = 97
4b

[35, 30.5, 20.0, 15.0, 10.5, 6.0, 5.0] => total après promotion = 122 - 20 - 6 = 96

4c

Pour avoir le prix après promotion de déstockage le plus bas possible, il faut trier le tableau dans l'ordre décroissant. On peut donc utiliser un algorithme de tri (tri par sélection, tri par insertion ou tri fusion)

Exercice 4

1a

racine => "Lea"
feuilles => "Marc", "Lea", "Claire", "Theo", "Marie", "Louis", "Anne" et "Kevin"

1b

```
def vainqueur(arb):
    return racine(arb)
```

1c

```
def finale(arb):
    f1 = gauche(arb)
    f2 = droit(arb)
    return [racine(f1), racine(f2)]
```

2a

```
from collections import deque

def occurrences(arb, nom):
    file = deque([arb])
    compteur =  0
    while len(file) > 0:
        x = file.popleft()
        if racine(x) == nom:
            compteur = compteur + 1
        if not est_vide(gauche(x)):
            file.append(gauche(x))
        if not est_vide(droit(x)):
            file.append(droit(x))
    return compteur
```

Il est aussi possible d'avoir une fonction récursive :

```
def occurrences(arb, nom):
  if est_vide(arb):
    return 0
  elif racine(arb) == nom:
      res = 1
  else:
    res = 0
  return res + occurrences(gauche(arb), nom) + occurrences(droit(arb),
nom)
```

2b

```
def a_gagne(arb, nom):
    return occurrences(arb,nom) > 1
```

3a

Les instructions proposées renvoient une valeur erronée dans le cas où le paramètre nom correspond au vainqueur du tournoi. En effet, si on considère l'arbre proposé à la question 1a, occurrences(arb, "Lea") renvoie 4 alors que Lea a joué seulement 3 matchs.

3b

```
def nombre_matchs(arb, nom):
    if vainqueur(arb)==nom :
        return occurrences(arb, nom) - 1
    else :
        return occurrences(arb, nom)
```

4

```
def liste_joueur(arb):
    if est_vide (arb):
        return []
    elif est_vide(gauche(arb)) and est_vide(droit(arb)) :
        return [racine(arb)]
    else :
        return  liste_joueurs(gauche(arb)) + liste_joueurs(droit(arb))
```

Exercice 5

1a

1b

```
def taille_file(F):
    t = 0
    ft = creer_file_vide()
    while not est_vide(F):
        t = t + 1
        enfiler(ft, defiler(F))
    while not est_vide(ft):
        enfiler(F, defiler(ft))
    return t
```

2

```
def former_pile(F):
    p = creer_pile_vide()
    pt = creer_pile_vide()
    while not est_vide(F):
        empiler(pt,defiler(F))
    while not est_vide(pt):
        empiler(p,depiler(pt))
    return p
```

3

```
def nb_elements(F, ele):
    nb = 0
    ft = creer_file_vide()
    while not est_vide(F):
        x = defiler(F)
        if x==ele:
            nb = nb + 1
        enfiler(ft, x)
    while not est_vide(ft):
        enfiler(F, defiler(ft))
    return nb
```

4

```
def verifier_contenu(F, nb_rouge, nb_vert, nb_jaune):
    return nb_elements(F, "rouge") <= nb_rouge and nb_elements(F,
"vert") <= nb_vert and nb_elements(F, "jaune") <= nb_jaune
```