Révisions_Correction

Exercice 1 : Réseaux

Partie A : Réseau

1

Protocole

2

a) élément A : Routeur
b) élément B : Switch

3

Matériel	Adresse IP	Masque	Passerelle
...	...	...	...
Poste 3	192.168.11.22	255.255.255.0	192.168.11.1

Partie B : Routage réseaux

1

Les adresses IP des réseaux directement connectés au routeur R1 (métrique égale à 0) sont : 10.0.0.0, 172.16.0.0 et 192.168.0.0

2

Adresse IP destination	Interface Machine ou Port
192.168.1.55	192.168.0.1
172.18.10.10	175.15.0.1

Partie B : Routage réseaux

1

Les adresses IP des réseaux directement connectés au routeur R1 (métrique égale à 0) sont : 10.0.0.0, 172.16.0.0 et 192.168.0.0

2

Adresse IP destination	Interface Machine ou Port
192.168.1.55	192.168.0.1
172.18.10.10	175.15.0.1

3. On ne considèrera que les chemins les plus courts.

Routeur destination	Métrique	Route
R2	0	R1-R2
R3	0	R1-R3
R4	1	R1-R2-R4
R5	1	R1-R3-R5
R6	1	R1-R3-R6
R7	2	R1-R2-R4-R7 (ou R1-R3-R6-R7)

Exercice 2 : fichiers CSV

1a

Un fichier CSV est un fichier au format “texte” permettant de “stocker” des données tabulées. Les données sont séparées par des virgules, d’où l’acronyme CSV : Comma Separated Values

1b

- prenom est de type string
- la réponse renvoyée par la fonction est aussi de type string

2a

```
import csv
```

2b

```
assert isinstance(prenom, str)
```

2c

```
def genre(prenom):
    liste_M = ['f', 'd', 'c', 'b', 'o', 'n', 'm', 'l', 'k','j', 'é',
'h', 'w', 'v', 'u', 't', 's', 'r','q', 'p', 'i', 'þ', 'z', 'x', 'ç',
'ö', 'ã','â', 'ï', 'g']
    liste_F = ['e', 'a', 'ä', 'ü', 'y', 'ë']
    if not isinstance(prenom, str):
        return "erreur, le prénom doit être une chaîne de caractères"
    if prenom[len(prenom)-1].lower() in liste_M :
        return "M"
    elif prenom[len(prenom)-1].lower() in liste_F :
        return "F"
    else :
        return "I"
```

3.

modification de la fonction genre (de la ligne 7 à la ligne 13) :

```
term = prenom[len(prenom)-2]+prenom[len(prenom)-1]
if term.lower() in liste_M2 :
      return "M"
elif term.lower() in liste_F2 :
      return "F"
else :
      return "I"
```

Exercice 3 : SGBD

1a

L'attribut *nom* de la table *licencies* ne peut pas servir de clé primaire, car il peut exister des homonymes et que la clé primaire doit être unique.

1b

L'attribut *id_licencie* peut jouer le rôle de clé primaire.

2a

Cette requête renvoie le nom et le prénom des licenciés qui jouent dans l'équipe des "- 12 ans".

2b

En utilisant * à la place de prenom, nom, on obtient l'ensemble des attributs.

2c

```
SELECT date
FROM matchs
WHERE equipe = 'Vétérans' AND lieu = 'Domicile'
```

3

```
INSERT INTO licencies
(id_licencie, prenom, nom, annee_naissance, equipe)
VALUES
(287, 'Jean', 'Lavenu', 2001, 'Hommes  2')
```

4

```
UPDATE Licencies
SET equipe = 'Vétérans'
WHERE prenom = 'Joseph' AND nom = 'Cuviller'
```

5

```
SELECT nom FROM licencies
JOIN Matchs ON licencies.equipe = matchs.equipe
WHERE adversaire = 'LSC' AND date = 19/06/2021
```

Exercice 4 : POO et Piles

1a

voici les 2 assertions dans la méthode `__init__` :

```
class Yaourt:
    def __init__(self,arome,duree):
        assert arome in ['fraise', 'abricot', 'vanille', 'aucun'], "Cet
arome est inconnu"
        assert duree > 0 and duree < 366, "la durée doit être comprise
entre 1 et  365"
        self.__arome = arome
        self.__duree = duree
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'
```

1b

Le genre associé à Mon_Yaourt sera `aromatise`

1c

Voici la **méthode** GetArome :

```
def GetArome(self):
        return self.__arome
```

2

```
def SetArome(self, arome):
       assert arome in ['fraise', 'abricot', 'vanille', 'aucun'], "Cet
arome est inconnu"
       self.__arome = arome
       self.__SetGenre(arome)
```

3a

```
def empiler(p, Yaourt):
    p.append(Yaourt)
    return p
```

3b

```
def depiler(p):
    return p.pop()
```

Remarque : il faudrait d'abord vérifier que la pile n'est ps vide avant de la dépiler.

3c

```
def estVide(p):
    return len(p)==0
```

3d

```
24
False
```

Exercice 5 : Piles

1

Avec cette méthode de mélange, on obtient :

```
['10', 'A', '9', 'R', '8', 'D', '7', 'V']
```

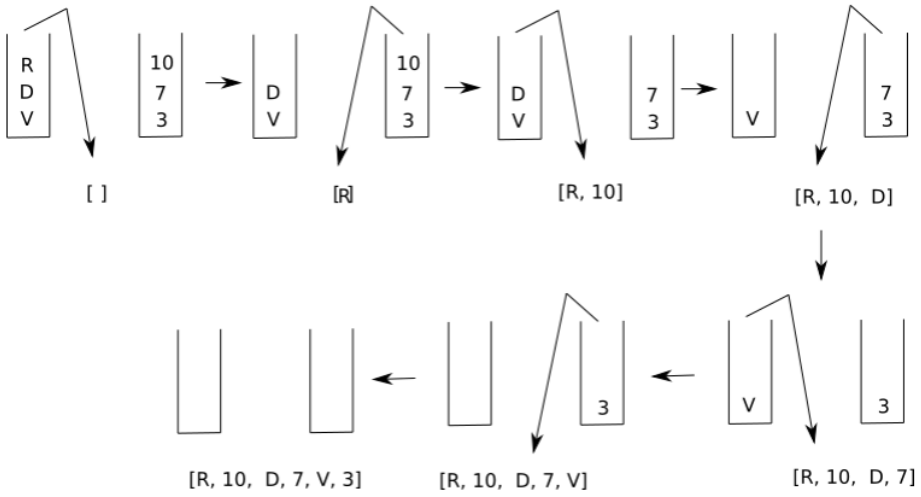
2

```
def liste_vers_pile(L):
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        p_temp.empiler(L[i])
    return p_temp
```

3

On obtient 3 ; 2 ; 1 et 6 ; 5 ; 4

4a



4b

```
def fusion(p1, p2):
    L = []
    while not p1.est_vide():
        L.append(p1.depiler())
        L.append(p2.depiler())
    return L
```

5

```
def affichage_pile(p):
    p_temp = p.copier()
    if p_temp.est_vide():
        print('_____')
    else:
        elt = p_temp.depiler()
        print('| ',elt,' |')
        affichage_pile(p_temp)
```