Métropole - septembre 2024 - sujet 1 (corrigé)

Exercice 1 (Programmation Python et bases de données)
Partie A

1. L'attribut `CP` peut être un INT ou un TEXT (STR)
2. La requête renvoie le nombre distinct de numéro de téléphone de la table `Agences`, soit 5
3. Il faut qu'il n'y ait pas deux agences avec le même numéro de téléphone.
4. On a la description suivante :`couple_voitures_agences(#id_agence : INT, #id_voiture : INT)`
5. La requête suivante convient :

```
INSERT INTO couple_voitures_agences VALUES(5, 2)
```

6. La requête suivante convient :

```
UPDATE couple_voitures_agences
SET id_agence = 2
WHERE id_voiture = 2
```

7. La requête suivante convient :

```
SELECT type, marque, Agence
FROM couple_voitures_agences
JOIN Agences ON couple_voitures_agences.id_agence = Agences.id_agence
JOIN Voitures ON couple_voitures_agences.id_voiture = Voitures.id_voiture
```

Partie B

8. La fonction suivante convient :

```
def insert_voiture(list_valeurs, id_agence):
    req1 = execute_requete_insert("""INSERT INTO Voitures
                                  ('marque', 'modele', 'kilometrage', 'nombre_place',
                                  'type', 'carburant')
                                  VALUES (list_valeurs[0], list_valeurs[1],
                                  list_valeurs[2], list_valeurs[3], list_valeurs[4],
                                  list_valeurs[5])""")
    req2 = execute_requete_insert("""INSERT INTO couple_voitures_agences
                                  VALUES(??,id_agence)""")
    return req1 AND req2
```

Remarque : Il y a un problème avec cette question : on ne connait pas l'identifiant de la voiture et on n'a aucun moyen de le récupérer...

9. Il faut bien vérifier la présence de tous les attributs dans le premier paramètre de la fonction. Il faut aussi vérifier le type de ces attributs et que l'id de l'agence existe bien.

Exercice 2 (Réseaux, protocoles de routage et graphes)

Partie A

1. Le portable P2 est situé dans le réseau `192.168.1.0` et les deux adresses IP `192.168.1.1` et `192.168.1.10` sont déjà utilisées. Comme l'adresse `192.168.1.255` est réservée, une adresse possible est `192.168.1.2`
2. En lisant les tables de routage, on déduit que le réseau L2 est connecté au routeur R9 via l'interface 3 et donc a pour adresse `192.168.18.0`. Comme deux adresses sont réservées (réseau et diffusion) et l'adresse `192.168.18.1` est déjà utilisée, on en déduit qu'il reste $256 - 3 = 253$ adresses possibles pour P3 et P4.

Partie B

3. L'implémentation suivante convient :

```
G = {'R1' : ['R2', 'R3', 'R4', 'R6'],
     'R2' : ['R1', 'R3', 'R5'],
     'R3' : ['R1', 'R2', 'R5', 'R6'],
     'R4' : ['R1', 'R6'],
     'R5' : ['R2', 'R3', 'R6', 'R7'],
     'R6' : ['R1', 'R3', 'R4', 'R5', 'R7', 'R8'],
     'R7' : ['R5', 'R6', 'R8', 'R9'],
     'R8' : ['R6', 'R7', 'R9'],
     'R9' : ['R7', 'R8']}
```

4. On a la table de routage suivante :

Destination	Suivant	Nombre de sauts
R2	R2	1
R3	R3	1
R4	R4	1
R6	R6	1
R5	R2	2
R8	R6	2
R7	R6	2
R9	R6	3

5. Le chemin suivant convient (3 sauts) : P1 → S1 → R1 → R6 → R8 → R9 → S2 → P3
6. L'implémentation suivante convient :

```
M = [[0,1,1,1,0,1,0,0,0],
     [1,0,1,0,1,0,0,0,0],
     [1,1,0,0,1,1,0,0,0],
     [1,0,0,0,0,1,0,0,0],
     [0,1,1,0,0,1,1,0,0],
     [1,0,1,1,1,0,1,1,0],
     [0,0,0,0,1,1,0,1,1],
     [0,0,0,0,0,1,1,0,1],
     [0,0,0,0,0,0,1,1,0]]
```

7. Le code suivant convient :

```
def degre(MATRICE):
    d = []
    for l in MATRICE :
        cpt = 0
        for v in l:
            cpt = cpt + v
        d.append(cpt)
    return d
```

8. L'appel `degre(M)` renvoie la liste `[4, 3, 4, 2, 4, 6, 4, 3, 2]`

9. Le graphe possède deux sommets de degré impair, il admet donc une chaîne eulérienne. Le robot va ainsi pouvoir parcourir l'ensemble du réseau en empruntant chaque fibre optique une et une seule fois.

Partie C

10. On a la route suivante pour un coût de 6 : P1 → S1 → R1 → R2 → R5 → R6 → R7 → R9 → S2 → P3

Exercice 3 (Pile et POO)

1. On a le tableau suivant :

Etage	Ecriture *Maya*	Valeur du « chiffre » de l'étage	Valeur dans la conversion
3		$1 \times 5 + 3 \times 1 = 8$	$8 \times 20^2 = 3\,200$
2		$2 \times 5 + 1 = 11$	$11 \times 20^1 = 220$
1		$3 \times 5 = 15$	$15 \times 20^0 = 15$

2. Compte-tenu du tableau précédent, on a bien $15 + 220 + 3200 = 3435$ en additionnant.
3. Les instructions suivantes conviennent :

```
M = Maya()
M.ajouter([0,0,3])
M.ajouter([0,1,2])
M.ajouter([0,3,1])
```

4. La méthode suivante convient :

```
def nbEtages(self) :
    return len(self.nombre)
```

5. La fonction suivante convient :

```
def valeurChiffre(L):
    return L[1] + 5 * L[2]
```

6. Le code suivant convient :

```
def MayaToDec(self):
    coeff = 20**(self.nbEtages() - 1)
    ch_Dec = 0
    while not self.estVide() :
        ch_Maya = self.retirer()
        ch_Dec = ch_Dec + (valeurChiffre(ch_Maya)) * coeff
        coeff = coeff // 20
    return ch_Dec
```

7. La fonction suivante convient :

```
def decompChiffre(n):
    if n == 0:
        return [1, 0, 0]
    return [0, n % 5, n // 5]
```

8. La fonction suivante convient :

```
def DecToMaya(n):
    M = Maya()
    for e in DecToVige(n):
        M.ajouter(decompChiffre(e))
    return M
```

9. La méthode suivante convient :

```
def multiplie(self):
    M = Maya()
    M.nombre = [[1,0,0]] + self.nombre
    return M
```

10. On a les résultats `([0, 4, 2], 0)` et `([1, 0, 0], 1)`
11. La méthode suivante convient :

```
def somme(self,maya2):
    if self.nbEtages() == maya2.nbEtages():
        r = 0
        res=Maya()
        for i in range(self.nbEtages()):
            s,r=mystere(self.nombre[i],maya2.nombre[i],r)
            res.ajouter(s)
        if r==1:
            res.ajouter([0,1,0])
        return res
```