Amérique du nord - mai 2022 - sujet 1 (corrigé)

Exercice 1 (SQL et bases de données)

1. (a) La relation `Sport` a pour clé primaire le couple d'attribut `(nomSport,nomSation)` et pour clé étrangère l'attribut `nomStation`.

   (b)
   - ⋆ Ccontrainte d'intégrité de domaine : l'attribut `prix` est de type nombre entier.
   - ⋆ Contrainte d'intégrité de relation : chaque couple d'attributs `(nomSport,nomSation)` doit être unique.
   - ⋆ Contrainte d'intégrité de référence : chaque valeur de l'attribut `nomStation` doit correspondre aux valeurs de l'attribut `nomStation` de la relation `Station`.

2. (a) Une requête d'insertion a été utilisée à la place d'une requête de mise à jour. L'entrée avec le couple `("planche à voile", "La tramontane catalane")` existe déjà dans la relation `Sport`, d'où l'erreur (rappel : chaque couple d'attributs `(nomSport, nomSation)` doit être unique).

   La requête correcte est la suivante :

```
L1 = resultats.queue()
L2 = L1.queue()
c1 = L2.tete()
```

   (b) La requête suivante convient :

```
INSERT INTO Sport VALUES ("plongée", "Soleil Rouge", 900)
```

3. (a) La requête suivante convient :

```
SELECT mail FROM Client
```

   (b) La requête suivante convient :

```
SELECT nomStation FROM Sport WHERE nomSport = "plongée"
```

4. (a) La requête suivante convient :

```
SELECT Station.ville, Station.nomStation
FROM Station
JOIN Sport ON Station.nomStation = Sport.nomStation
WHERE Sport.nomSport = "plongée"
```

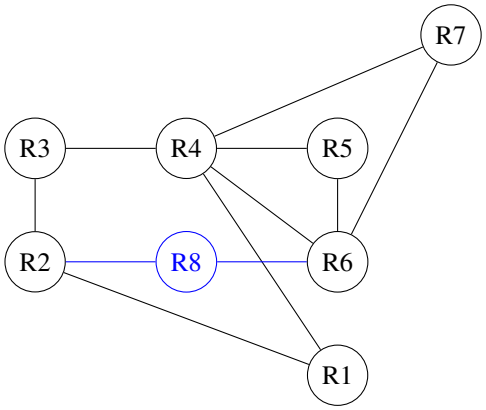
   (b) La requête suivante convient :

```
SELECT COUNT(*)
FROM Sejour
JOIN Station ON Sejour.nomStation = Station.nomStation
WHERE Station.region = "Corse"
```

Exercice 2 (Réseaux et protocoles de routage)

1. Le paquet de données utilise le chemin suivant : R2 – R1 – R4 – R7
   L'accusé de réception utilise le chemin suivant : R7 – R4 – R3 – R2
2. (a) Dans le cas d'une panne du routeur R4 le groupe de routeur (R1, R2, R3) n'est plus capable d'atteindre le groupe de routeur (R5, R6, R7)

   (b) On pourrait, entre autre, établir une liaison entre le routeur R1 et R6.
3. On a le graphe suivant :



   (a) La table de routage de R8 est la suivante :

Destination	Lien	Distance
R1	R2	2
R2	R2	1
R3	R2	2
R4	R6	2
R5	R6	2
R6	R6	1
R7	R6	2

   (b) La table de routage de R2 est la suivante :

Destination	Lien	Distance
R1	R1	1
R3	R3	1
R4	R1	2
R5	R3	3
R6	R8	2
R7	R1	3
R8	R8	1

4. (a) La bande passante du Fast Ethernet est de 10^8 b/s soit 100 Mb/s.
   Le coût du réseau Ethernet est de $10^8/10^7 = 10$.

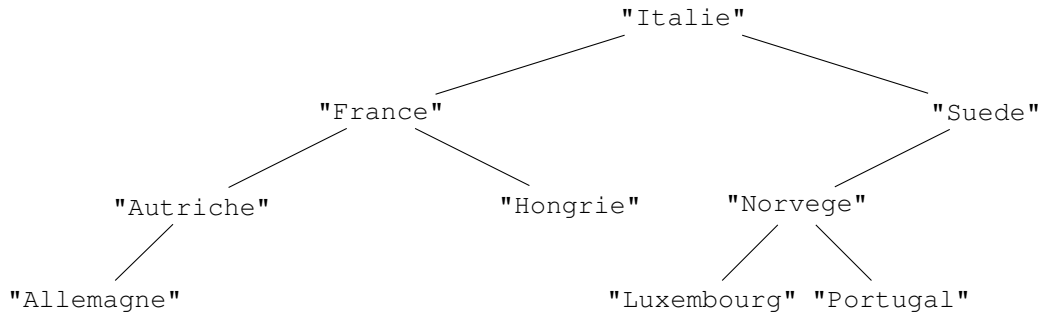
   (b) Le chemin le moins coûteux est R2 – R3 – R4 – R7 – R6 – R5 avec un coût égale à $65 + 10 + 1 + 1 + 10 = 87$. Tous les autres trajets entre R2 et R5 ont un coût supérieur (par exemple R2 – R1 – R4 – R5 a un coût de $49 + 65 + 49 = 163$).

Exercice 3 (Arbres binaires de recherche)

1. (a) La hauteur de l'arbre est de 3.

   (b) La valeur booléenne de l'expression est `True`.

   (c) On a l'arbre suivant :



2. Le parcours en largeur de l'arbre donne :
   `"Italie" - "France" - "Suede" - "Autriche" - "Hongrie" - "Norvege"`

3. La fonction suivante convient :

```
def recherche(arb, val):
    """la fonction renvoie True si val est dans l'arbre
    et False dans le cas contraire"""
    if est_vide(arb):
        return False
    if val == racine(arb):
        return True
    if val < racine(arb):
        return recherche(gauche(arb), val)
    else :
        return recherche(droit(arb), val)
```

4. La fonction suivante convient :

```
def taille(arb):
    if est_vide(arb):
        return 0
    else:
        return 1 + taille(gauche(arb)) + taille(droit(arb))
```

Exercice 4 (Chaînes de caractères, tableaux et programmation)

1. (a) La chaîne de caractères est automatique un palindrome dès que sa longueur est 0 ou 1. C'est donc la Proposition 3 qui est la bonne.

   (b)
   - Les variables `txt[0]` et `txt[taille-1]` contiennent respectivement les valeurs `b` et `r`.
   - La variable `interieur` contient la valeur `onjou`

2. On peut tester un cas où la fonction doit renvoyer `True` (exemple : `palindrome("BOB")`) et un cas où la fonction doit renvoyer `False` (exemple : `palindrome("BONJOUR")`).

3. La fonction suivante convient :

```
def palindrome(txt):
    taille = len(txt)
    if taille < 2:
        return True
    i = 0
    j = taille - 1
    while i < j :
        if txt[i] != txt[j]:
            return False
        i = i + 1
        j = j - 1
    return True
```

4. (a) La fonction suivante convient :

```
def complementaire(txt):
    c = ""
    for l in txt :
        if l == "A":
            c = c + "T"
        if l == "T":
            c = c + "A"
        if l == "G":
            c = c + "C"
        if l == "C":
            c = c + "G"
    return c
```

   (b) La chaine `"GATCGT"` n'est pas palindromique, car la concaténation donne `"GATCGTCTGCA"` qui n'est pas un palindrome.

   (c) La fonction suivante convient :

```
def est_palindromique(txt):
    comp = complementaire(txt)
    conc = txt+comp
    return palindrome(conc)
```

Exercice 5 (Files et tableaux)

1. (a) Les éléments 15, 17 et 14 ont été enfilés dans cette ordre, donc c'est la Proposition 2 qui est correcte.

   (b) Les instructions suivantes conviennent :

```
f = creer_file_vide()
enfiler(f, 15)
enfiler(f, 17)
enfiler(f, 14)
```

2. La fonction suivante convient :

```
def longueur_file(F):
    G = creer_file_vide()
    n = 0
    while not(est_vide(F)):
        v = defiler(F)
        n = n + 1
        enfiler(G, v)
    while not(est_vide(G)):
        v = defiler(G)
        enfiler(F, v)
    return n
```

3. La fonction suivante convient :

```
def variations(F):
    taille = longueur_file(F)
    if taille == 1 :
        return []
    else:
        tab = [0 for k in range(taille - 1)]
        element1 = defiler(F)
        for i in range(taille - 1):
            element2 = defiler(F)
            tab[i]=element2 - element1
            element1 = element2
    return tab
```

4. La fonction suivante convient :

```
def nombre_baisses(tab):
    mini = tab[0]
    nbr = 0
    for v in tab:
        if v < 0:
            nbr = nbr + 1
        if v < mini:
            mini = v
    if nbr == 0:
        return (0,0)
    else:
        return (nbr, mini)
```