Amérique du nord - mai 2022 - sujet 1

Exercice 1 (SQL et bases de données - 4 points)
L'énoncé de cet exercice utilise les mots du langage SQL suivants :

`SELECT`, `FROM`, `WHERE`, `JOIN`, `INSERT INTO`, `VALUES`, `UPDATE`, `SET`, `COUNT`

On rappelle qu'en SQL, la fonction d'agrégation `COUNT` permet de compter le nombre d'enregistrements dans une table.

L'entreprise `Vacances autrement` propose des séjours d'une durée d'une semaine dans différentes stations situées en France. Les séjours sont organisés autour d'une seule activité sportive. Par exemple, la station `La tramontane catalane` située à Leucate propose une formule autour de la planche à voile. Les clients pourront durant toute la durée du séjour pratiquer ce sport : le matériel est fourni et deux sessions encadrées par des moniteurs diplômés sont organisées chaque journée. Cette même station propose aussi des séjours pour les amateurs de kitesurf.
Les réservations se font par l'intermédiaire d'un site internet : chaque cloent complète un formulaire avec ses données personnelles (nom, prénom, etc.) et choisit le séjour souhaité.
Ce site s'appuie sur une base de données relationnelle. Elle contient notamment les relations `Station`, `Sport`, `Client` et `Sejour`.

Voici le schéma relationnel où les clefs primaires sont soulignées et où les clefs étrangères sont suivies du symbole `#` :

```
Station (nomStation, ville, region)
Sport (nomSport, nomStation#, prix)
Client (cID, nom, prenom, mail)
Sejour (cID, semaine, annee, nomStation#, nomSport#)
```

- Les différentes stations sont stockées dans la relation `Station`. Chaque station est identifiée par un nom caractéristique (`nomStation`). Les attributs `ville` et `region` permettent de décrire la localisation de chacune des stations.
- La relation `Sport` fait le lien entre les stations et les activités sportives qui y sont proposées. Les trois attributs sont `nomSport`, `nomStation` et `prix`. L'attribut `prix` est un nombre entier : il correspond au montant plein tarif (en euros) du séjour d'une semaine dans la station avec le sport associé.
- La relation `Client` caractérise les clients de l'entreprise : ils sont identifiés par un nombre entier (`cID`) et décrits par leur nom, prénom et adresse mail.
- La relation `Sejour` permet de lister les séjours auxquels les clients ont participé. Les deux attributs `semaine` et `annee` sont des nombres entiers : ils permettent d'identifier le moment où a été effectué chaque séjour (`semaine` est le numéro de la semaine pendant laquelle le séjour a été réalisé).
  Exemple : un séjour ayant été effectué la semaine numéro 12 de l'année 2020 correspond à `semaine = 12` et à `annee = 2020`.

On donne aussi un extrait des trois premières lignes de chacune de ces relations :

`Station`

nomStation	ville	region
La tramontane catalane	Leucate	Occitanie
La baie sauvage	La Torche	Bretagne
La pinède	Calvi	Corse

`Sport`

nomSport	nomStation	prix
planche à voile	La tramontane catalane	1200
kitesurf	La tramontane catalane	1100
plongée	La baie sauvage	950

`Client`

cID	nom	prenom	mail
1	GENEREUX	Eric	eric.genereux@mail.fr
2	PIERRE	Daniel	daniel.pierre@mail.fr
3	JOLY	Emilie	emilie.joly@mail.fr

`Sejour`

cID	semaine	annee	nomStation	nomSport
1	26	2020	La tramontane catalane	planche à voile
1	38	2020	La tramontane catalane	planche à voile
2	33	2020	La baie sauvage	plongée

1. (a) Donner la clé primaire et les éventuelles clés étrangères de la relation `Sport`.

   (b) Citer une contrainte d'intégrité de domaine, puis une contrainte d'intégrité de relation et enfin une contrainte d'intégrité de référence que doivent respecter les données de la relation `Sport`.

2. (a) Le tarif du séjour à la station `La tramontane catalane` basé sur la planche à voile passe de 1200 euros à 1350 euros. La requête SQL suivante a été utilisée pour la mise à jour du tarif :

```
INSERT INTO Sport VALUES ("planche à voile", "La tramontane catalane", 1350);
```

   Cette requête a été rejetée et la mise à jour n'a pas été effectuée. Après avoir expliqué pourquoi cette requête a été refusée, proposer une requête SQL qui permettra de modifier le tarif de ce séjour.

   (b) Une nouvelle station vient d'être référencée. Son nom est `Soleil Rouge`. Elle est située à Bastia en Corse. Des séjours d'une semaine y seront organisés. Le sport pratiqué sera la plongée au tarif de 900 euros.
   Ecrire les requêtes SQL permettant d'insérer ces nouvelles données dans la base.

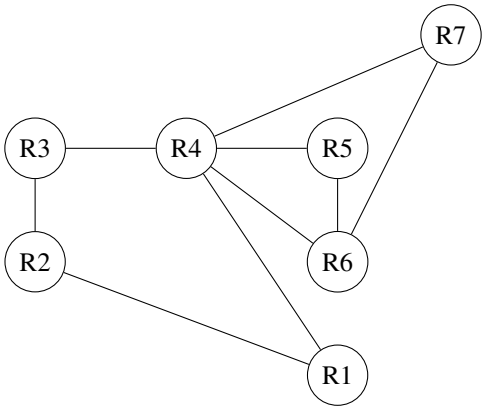
3. (a) L'agence souhaite envoyer un mail d'information à ses clients pour leur présenter cette nouvelle possibilité de séjour mise en place à Bastia. Ecrire une requête SQL permettant d'obtenir l'adresse mail de tous les clients.

   (b) Un client qui pratique la plongée souhaite réserver un séjour. Ecrire une requête SQL permettant d'obtenir le nom de toutes les stations où l'on peut pratiquer la plongée.

4. (a) Pour faire son choix, le client souhaiterait connaitre les villes où sont situées les stations dans lesquelles il pourra pratiquer la plongée. Ecrire une requête SQL permettant d'obtenir le nom des villes ainsi que le nom des stations où l'on peut pratiquer la plongée.

   (b) Ecrire une requête SQL permettant de déterminer le nombre total de séjours effectués en Corse durant l'année 2020.

Exercice 2 (Réseaux et protocoles de routage - 4 points)
On considère le réseau suivant composé de sept routeurs :



On donne ci-dessous les tables de routage préalablement construites avec le protocole RIP (Routing Information Protocole). Le protocole RIP permet de construire les tables de routage des différents routeurs en indiquant, pour chaque routeur, la distance, en nombre de sauts, qui le sépare d'une autre routeur.

Table de routage de R1

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

Table de routage de R2

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

Table de routage de R3

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

Table de routage de R4

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

Table de routage de R5

Destination	Lien	Distance
R1	R4	2
R2	R4	3
R3	R4	2
R4	R4	1
R6	R6	1
R7	R6	2

Table de routage de R6

Destination	Lien	Distance
R1	R4	2
R2	R4	3
R3	R4	2
R4	R4	1
R5	R5	1
R7	R7	1

Table de routage de R7

Destination	Lien	Distance
R1	R4	2
R2	R4	3
R3	R4	2
R4	R4	1
R5	R4	2
R6	R6	1

1. Le routeur R2 doit envoyer un paquet de données au routeur R7 qui en accuse réception. Déterminer le chemin parcouru par le paquet de données, ainsi que celui parcouru par l'accusé de réception.
2. (a) Indiquer la faiblesse que présente ce réseau en cas de panne du routeur R4.
   (b) Proposer une solution pour y remédier.
3. Dans cette question uniquement, on décide de rajouter un routeur R8 qui sera relié aux routeurs R2 et R6.
   (a) Donner une table de routage pour R8 qui minimise le nombre de saut.
   (b) Donner une nouvelle table de routage de R2.

4. **Pour la suite de l'exercice, on considèrera le réseau sans le routeur R8.**
Il a été décidé de modifier les règles de routage de ce réseau en appliquant dorénavant le protocole de routage OSPF qui prend en compte la bande passante.
Ce protocole attribue un coût à chaque liaison afin de privilégier le choix de certaines routes plus rapides. Plus le coût est faible, plus le lien est intéressant.
Le coût d'une liaison est calculé par la formule :

$$\text{coût} = \frac{10^8 \text{ bit/s}}{\text{bande passante du lien en bit/s}}.$$

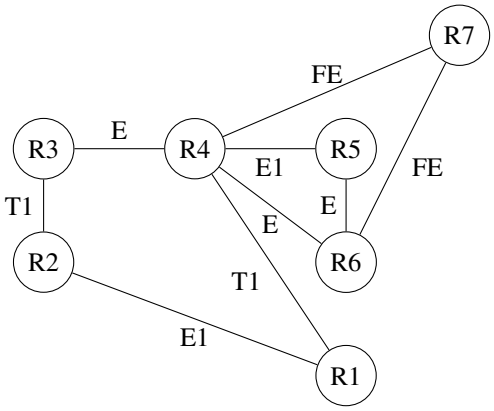
Voici le tableau référençant les coûts des liaisons en fonction du type de liaison entre deux routeurs :

Type de liaison	Bande passante	Coût
FastEthernet (FE)	?	1
Ethernet (E)	10 Mb/s	?
(E1)	2,048 Mb/s	49
(T1)	1,544 Mb/s	65

On rappelle que 1 Mb/s $= 1\,000$ kb/s $= 10^6$ bit/s.

(a) Déterminer la bande passante du FastEthernet (FE) et justifier que le coût du réseau de type Ethernet (E) est de 10.

(b) On précise sur le graphe ci-dessous les types de liaison dans notre réseau :



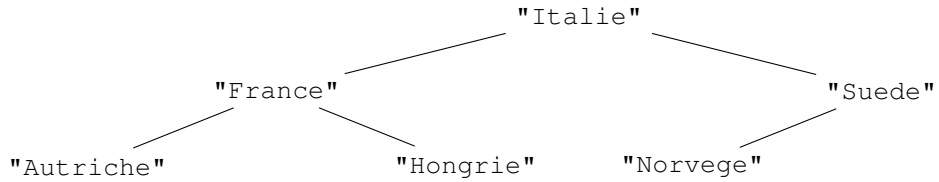
Le coût d'un chemin est la somme des coûts des liaisons rencontrées. Donner, en justifiant, le chemin le moins coûteux pour relier R2 à R5. Préciser le coût de ce chemin.

Exercice 3 (Arbres binaires de recherche - 4 points)

Les arbres binaires de recherche considérés ici sont des arbres binaires où les nœuds désignent des chaînes de caractères et pour lesquelles la valeur de chaque nœud est supérieure à celles des nœuds de son enfant gauche, et inférieure à celles des nœuds de son enfant droit.
La relation d'ordre notée < est ici la relation d'ordre alphabétique.

Dans cet exercice, on utilisera la convention suivante : la hauteur d'un arbre binaire ne comportant qu'un nœud est 1.
Dans cet exercice, les arbres binaires de recherche ne contiennent que des noms de pays tous distincts.

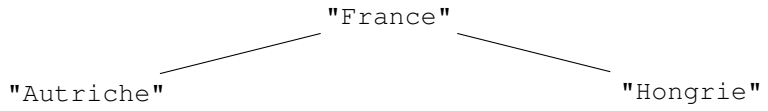
On considère l'arbre binaire de recherche suivant :



1. (a) Donner, sans justification, la hauteur de cet arbre.
   (b) Donner, sans justification, la valeur booléenne de l'expression `"Allemagne" < "Portugal"`.
   (c) Recopier l'arbre après l'ajout de `"Allemagne"`, de `"Portugal"` et de `"Luxembourg"` dans cet ordre.

Pour les question 2, 3 et 4, on traite l'arbre initial, donc sans l'ajout de `"Allemagne"`, `"Portugal"` et `"Luxembourg"`.

2. On souhaite parcourir l'arbre. Indiquer l'ordre de visite des nœuds lors d'un parcours en largeur.
3. On souhaite écrire une fonction pour déterminer si le nom d'un pays est dans l'arbre. On dispose pour cela de :
   - la fonction `est_vide` qui prend en paramètre un arbre `arb`. Cette fonction renvoie `True` si l'arbre `arb` est vide, `False` sinon ;
   - la fonction `gauche` qui prend en paramètre un arbre `arb` et renvoie son sous-arbre gauche.
     Exemple : si `A` est notre arbre initial, alors `gauche(A)` renvoie l'arbre



   - la fonction `droite` qui prend en paramètre un arbre `arb` et renvoie son sous-arbre droit.
     Exemple : si `A` est notre arbre initial, alors `droite(A)` renvoie l'arbre

   - la fonction `racine` qui prend en paramètre un arbre `arb` et qui renvoie la valeur de la racine de l'arbre.
     Exemple : si `A` est notre arbre initial, alors `racine(A)` renvoie `"Italie"`.

   Recopier, en complétant les lignes 2, 6, 7 et 10, la fonction `recherche` donnée ci-dessous et écrite en Python. Cette fonction prend en paramètre un arbre `arb` et une valeur `val`. L'appel `recherche(arb,val)` renvoie un booléen : `True` si la valeur `val` est dans l'arbre `arb`, `False` sinon.

```
1  def recherche(arb, val):
2      """......................................................................"""
3      if est_vide(arb):
4          return False
5      if val == racine(arb):
6          return .....................
7      if val ..................... :
8          return recherche(gauche(arb), val)
9      else:
10         return .....................................
```

4. Ecrire une fonction récursive `taille` permettant de déterminer le nombre de pays présent dans un arbre. Cette fonction prendra en paramètre un arbre `arb` et renverra un entier.

Exercice 4 (Chaînes de caractères, tableaux et programmation - 4 points)

On appelle palindrome un texte dont l'ordre des lettres reste le même, qu'on le lise de la droite vers la gauche ou de la gauche vers la droite.
Par exemple, un mot à une lettre est un palindrome, `"BOB"` est un palindrome, tout comme `"LAVAL"`.
Le mot `""` est considéré comme un palindrome.

On souhaite programmer une fonction `palindrome` qui prend en paramètre une chaîne de caractères `txt`. Cette fonction renvoie `True` si la chaîne de cractères `txt` est un palindrome, `False` sinon.

On rappelle que :

- la fonction `len` prend une chaîne de caractères en paramètre et renvoie sa longueur, c'est-à-dire le nombre de caractères constituant la chaîne de caractères.
  Exemple : `len("bonjour")` renvoie `7`.
- Si `txt` est une chaîne de caractères, alors `txt[i]` est le caractère de `txt` d'indice `i`. Attention, le premier caractère a pour indice `0`.
  Exemple : si `txt = "bonjour"`, alors `txt[2]` désigne le caractère `"n"`.

1. On donne ci-dessous une implémentation récursive incomplète pour la fonction `palindrome`.
   L'opérateur + à la ligne 8 permet de concaténer deux chaînes de caractères.
   Exemple : si `txt1 = "bon"` et `txt2 = "jour"`, alors l'instruction `txt1 + txt2` renvoie la chaîne de caractères `"bonjour"`.

```
1 def palindrome(txt):
2     """ Str -> Bool """
3     ........................
4     ........................
5     taille = len(txt)
6     interieur = ""
7     for i in range(1, taille - 1):
8         interieur = interieur + txt[i]
9     return (txt[0] = txt[taille - 1]) and (palindrome(interieur))
```

   (a) Choisir parmi les propositions ci-dessous celle qui convient pour compléter la fonction `palindrome` (lignes 3 et 4) :

   Proposition 1 :

```
if len(txt) <= 2:
    return True
```

   Proposition 2 :

```
if len(txt) < 2:
    return False
```

   Proposition 3 :

```
if len(txt) < 2:
    return True
```

   Proposition 4 :

```
if len(txt) <= 2:
    return False
```

   (b) Lors de l'appel de `palindrome("bonjour")`, indiquer les valeurs, à la ligne 9, des variables `txt[0]`, `txt[taille - 1]` et `interieur`.

2. Proposer deux tests pour cette fonction qui permettent de tester deux cas de figures différents. Justifier ce choix.
3. Ecrire une version non récursive de la fonction `palindrome`.
4. On étudie dans cet question des chaînes de caractères utilisant uniquement les lettres `"A"`, `"T"`, `"C"` et `"G"`.
   Exemples : `"AA"`, `"CAT"` et `"CCGATACG"`.

   On associe à chacune de ces lettres une autre lettre, appelée lettre complémentaire, selon le tableau suivant :

Lettre	`"A"`	`"T"`	`"C"`	`"G"`
Lettre complémentaire	`"T"`	`"A"`	`"G"`	`"C"`

   On obtient le complémentaire d'un mot en remplaçant chacune de ses lettres par sa lettre complémentaire.
   Exemple : Le complémentaire de `"GAATTC"` est `"CTTAAG"`.

(a) Ecrire une fonction Python, nommée `complementaire`, qui prend en paramètre une chaîne de caractères `txt` écrite uniquement avec les lettres `"A"`, `"T"`, `"C"` et `"G"`. Cette fonction renvoie la chaîne de caractères complémentaire de `txt`. Exemple : l'appel `complementaire("GAATTC")` renvoie `"CTTAAG"`.

(b) Une chaîne de caractères `txt` est dite palindromique si la concaténation de `txt` avec son complémentaire est un palindrome. Exemples :

- `"GAATTC"` est palindromique car `"GAATTC" + "CTTAAG" = "GAATTCCTTAAG"` est un palindrome.
- `"GAAT"` n'est pas palindromique car `"GAAT" + "CTTA" = "GAATCTTA"` n'est pas un palindrome.

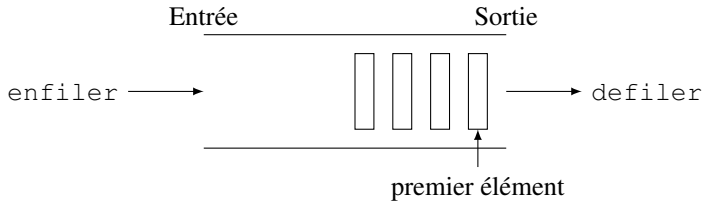
Déterminer si la chaîne de caractères `"GATCGT"` est palindromique.

(c) Ecrire une fonction `est_palindromique` prenant comme paramètre une chaîne de caractères `txt`. Cette fonction doit renvoyer `True` s'il s'agit d'une séquence palindromique, `False` sinon.

Exercice 5 (Files et tableaux - 4 points)

L'objectif de cet exercice est de travailler sur les températures relevées par un station météorologique. Les données sont enregistrées une fois par jour, à la même heure, et traitées dans l'ordre dans lequel elles arrivent.

On choisit d'utiliser une file. On rappelle qu'une file est une structure de données abstraites fondée sur le principe « premier arrivé, premier sorti ».



On munit la structure de données `File` des quatre fonctions primitives définies ci-dessous :

Structure de données abstraite : `File`
Utilise : Elément, Booléen

Opérations :
• `creer_file_vide` : $\varnothing \rightarrow$ `File`
`creer_file_vide()` renvoie une file vide
• `est_vide` : `File` $\rightarrow$ `Booléen`
`est_vide(F)` renvoie `True` si la file `F` est vide, `False` sinon
• `enfiler` : `File, Elément` $\rightarrow \varnothing$
`enfiler(F, element)` ajoute `element` en entrée de la file `F`
• `defiler` : `File` $\rightarrow$ `Elément`
`defiler(F)` renvoie l'élément en sortie de la file `F` (premier élément) en le retirant de la file `F`

1. Les températures relevées ont été 15, puis 17, puis 14.

   (a) Parmi les quatre propositions suivantes, indiquer celle qui représente correctement cette file :

   Proposition 1 : Entrée `15 17 14` Sortie — Le premier élément est `14`

   Proposition 2 : Entrée `14 17 15` Sortie — Le premier élément est `15`

   Proposition 3 : Entrée `15 17 14` Sortie — Le premier élément est `15`

   Proposition 4 : Entrée `14 17 15` Sortie — Le premier élément est `14`

   (b) En utilisant les fonctions primitives précédentes, donner les instructions permettant de créer cette file.

2. On appelle longueur d'une file le nombre d'éléments qu'elle contient.
   La fonction `longueur_file` prend en paramètre une file `F` et renvoie sa longueur `n`.
   Après appel de cette fonction, la file `F` doit avoir retrouvé son état d'origine.

   Exemple : si `F =` `10 10 12 12`, alors `longueur_file(F)` renvoie `4`.

   Recopier et compléter le programme Python suivant qui permet d'implémenter la fonction `longueur_file` (les trois points `...` peuvent correspondre à une ou plusieurs lignes de programme).

```
def longueur_file(F):
    """File -> Int"""
    G = creer_file_vide() # file temporaire
    n = 0 # initialisation du nombre d'éléments
    while not(est_vide(F)):
        ...
    while not(est_vide(G)): # reconstruction de la file initiale
        ...
    return ...
```

3. On s'intéresse à la variation de température d'un jour sur l'autre.
Par exemple, lorsque les températures relevées sont dans l'ordre d'arrivée 15, 17 et 14, les variations sont 2 et -3.
Recopier et compléter le programme Python implémentant la fonction `variations` qui prend en paramètre une file non vide `F` et qui renvoie le tableau `tab` contenant les variations successives, ou un tableau vide si la file `F` ne contient qu'une seule température. Il n'est pas demandé ici que la file `F` retrouve son état d'origine après appel de la fonction `variations`.
Exemple : si `F` est la file qui contient dans l'ordre des relevés les valeurs `15`, `17` et `14`, alors l'appel `variations(F)` renvoie `[2, -3]`.
Dans le code de la fonction, les trois points `...` peuvent correspondre à une ou plusieurs lignes de programme.

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

4. Ecrire une fonction `nombre_baisses` qui prend en paramètre un tableau `tab`, non vide, des variations des températures et qui renvoie un p-uplet contenant le nombre de jours où la température a baissé par rapport au jour précédent (soit le nombre de valeurs strictement négatives de `tab`), ainsi que la baisse journalière la plus importante (soit la valeur minimale de `tab`).
S'il n'y a aucune baisse (toutes les valeurs de `tab` sont positives), la fonction renvoie le p-uplet `(0,0)`.
Exemples :
   - `nombre_baisses([1, -4, 2, -1, 3])` renvoie `(2, -4)`.
   - `nombre_baisses([1, 5, 3, 1])` renvoie `(0, 0)`.