BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

Décembre 2022 – Bac Blanc

N.S.I.
Numérique et Sciences Informatiques

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice n'est pas autorisé

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.
Ce sujet comporte 9 pages numérotées de 1 à 9

Chaque exercice est noté sur 4 points

Exercice 1 : *(4 points)*

Thèmes abordés : bases de données et langage SQL.

On souhaite gérer un club de tennis en ligne avec la possibilité de réserver un terrain à un créneau horaire. Le site ne gère que des réservations pour des matchs en simple. Voici la structure de la base de données :

- Relation contenant l'ensemble des joueurs du club avec leurs identifiants.

joueurs				
id_joueur	**nom_joueur**	**prenom_joueur**	**login**	**mdp**
1	Dupont	Alice	alice	1234
2	Durand	Belina	belina	5694
3	Caron	Camilia	camilia	9478
4	Dupont	Dorine	dorine	1347

- Relation précisant les matchs joués.

matchs					
id_match	**date**	**id_creneau**	**id_terrain**	**id_joueur1**	**id_joueur2**
1	2020-08-01	2	1	1	4
2	2020-08-01	3	1	2	3
3	2020-08-02	6	2	1	3
4	2020-08-02	7	2	2	4
5	2020-08-08	3	3	1	2
6	2020-08-08	5	2	3	4

- Relation précisant les différents terrains.

terrains		
id_terrain	**nom_terrain**	**surface**
1	stade	terre battue
2	gymnase	synthétique
3	hangar	terre battue

- Relation précisant les créneaux réservables.

creneaux	
id_creneau	**plage_horaire**
1	8h-9h
2	9h-10h
3	10h-11h
4	11h-12h
5	12h-13h
6	13h-14h
7	14h-15h
8	15h-16h
9	16h-17h
10	17h-18h
11	18h-19h
12	19h-20h

1. Clés primaires/étrangères :
   a. Donner la clé primaire de la relation `matchs`.
   b. La relation `matchs` a-t-elle une ou des clés étrangères ? Si oui quelles sont-elles ?
2. Par lecture et analyse des relations de la base de donnée.
   a. Déterminer le jour et la plage horaire du match entre Durand Belina et Caron Camilia.
   b. Déterminer le nom des deux joueurs qui sont les seuls à avoir joué dans le hangar.
3. Requêtes en langage SQL (On pourra s'aider de l'annexe)
   a. Ecrire une requête qui renvoie les prénoms des joueurs dont le nom est 'Dupont'.
   b. Ecrire une requête qui modifie le mot de passe de Dorine Dupont, son nouveau mot de passe étant 1976.
4. Ecrire une requête permettant d'ajouter le nouveau membre « Zora MAGID » dont le login est « zora » et le mot de passe 2021.
5. Ecrire une requête qui renvoie les jours où Alice joue.

Exercice 2 : *(4 points)*

Cet exercice porte sur les structures de données (dictionnaires).

La cryptographie est un ensemble de techniques permettant de chiffrer un message.

Une technique de cryptographie consiste à mélanger les lettres d'un alphabet et à réécrire le message avec ces permutations. En Python, on peut créer un dictionnaire dans lequel les clés sont les lettres de l'alphabet et les valeurs sont celles de l'alphabet mélangé.

Par exemple, si l'alphabet contient les 4 lettres A, B, C et D, et si le dictionnaire de l'alphabet mélangé est `alpha ={"A": "B", "B": "D", "C": "A", "D": "C"}`, la chaîne de caractères `"BAC"` sera chiffrée `"DBA"`.

Un tel dictionnaire sera appelé **dictionnaire de chiffrement**.

1. On souhaite chiffrer un message écrit avec l'alphabet `A, B, C, D, E, F, G` à l'aide du dictionnaire `alpha ={"A":"B", "B":"D", "C":"A", "D":"C", "E":"F", "F":"G", "G":"E"}`

   a. Quelle est la valeur associée à la clé `"D"` ? En Python, comment l'obtenir ?
   b. Chiffrer la chaîne de caractères `"BAGAGE"` avec le dictionnaire `alpha`.

2. On considère qu'un mot est une chaîne de caractères (un objet de type `str`) écrite uniquement avec les 26 lettres de l'alphabet en majuscule. Par exemple, `"ARBRE"` est un mot et `"L'ARBRE !"` n'est pas un mot à cause des caractères : `"'", " "`(espace) et `"!"`.
   Écrire une fonction `chiffrer(mot, alpha)` qui prend en paramètres `mot` un mot et `alpha` un dictionnaire de chiffrement, et qui renvoie une chaîne de caractères chiffrée avec le dictionnaire de chiffrement `alpha`.

3. On souhaite déchiffrer un mot chiffré avec cette méthode.

   a. Si un mot est chiffré avec le dictionnaire de chiffrement `alpha ={"A":"B", "B":"D", "C":"A", "D":"C", "E":"F", "F":"G", "G":"E"}`, donner un dictionnaire permettant de le déchiffrer.

   b. Ecrire une fonction en Python appelée `dico_dechiffrement(dico)` qui prend en paramètre `dico` un dictionnaire de chiffrement et qui renvoie un dictionnaire permettant le déchiffrement. On pourra s'inspirer du code incomplet ci-dessous ou proposer une autre solution :

```
def dico_dechiffrement(dico):
    nouveau = {}
    for lettre in dico :
        code = dico[……]
        nouveau[……] = ……
    return nouveau
```

c. Ecrire une fonction `dechiffre(mot, dico)` qui reçoit un mot chiffré et un dictionnaire de chiffrement et renvoie le mot décodé. On utilisera les fonctions écrites dans les questions précédentes.

4. On souhaite à présent créer un dictionnaire de chiffrement. Ecrire une fonction `dico_chiffrement(alphabet)` qui prend en paramètre `alphabet` un tableau de lettres et qui renvoie un dictionnaire de chiffrement dont les clés sont les lettres du tableau `alphabet` et les valeurs sont les lettres du tableau `alphabet` mélangées.

On pourra utiliser la fonction `shuffle` du module `random` qui mélange en place un tableau. Par exemple, on a :

```
>>> tab = ["A", "B", "C", "D"]
>>> shuffle(tab)
>>> tab
["B", "A", "D", "C"]
```

Exercice 3 : *(4 points)*

Cet exercice porte sur la programmation en Python, la récursivité et la méthode "diviser pour régner".

Une ligne polygonale est constituée d'une liste ordonnée de points, appelés sommets, joints par des segments. L'algorithme de Douglas-Peuker permet de simplifier une ligne polygonale en supprimant certains de ses sommets. L'effet de l'algorithme appliqué aux lignes polygonales du contour de la France métropolitaine est illustré ci-dessous.

Avant application de l'algorithme *Après application de l'algorithme*

On implémentera cet algorithme dans la dernière question de l'exercice. Pour cela nous allons d'abord implémenter des fonctions auxiliaires.
On suppose dans la suite que les sommets sont des points du plan dont les coordonnées (x, y) dans un repère orthonormé fixé sont représentées par des tuples de longueur 2.

1. La distance qui sépare deux points A et B de coordonnées (x_A, y_A) et (x_B, y_B) est donnée par la formule $\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$.
   On rappelle que la fonction `sqrt` du module `math` de Python renvoie la racine carrée d'un nombre positif ou nul.
   a. Écrire une instruction qui permet d'importer la fonction `sqrt` du module `math`.
   b. Supposant l'import réalisé, écrire une fonction `distance_points(a,b)` qui prend en argument deux tuples `a` et `b` représentant les coordonnées de deux points et renvoie la distance qui les sépare.

2. On dispose d'une fonction `distance_point_droite(p, a, b)` qui prend en argument les tuples représentant les coordonnées respectives des points P, A et B, et qui renvoie la distance du point P à la droite (AB). L'exécution de cette fonction produit une erreur dans le cas où les points A et B sont égaux.
   À l'aide des fonctions `distance_points` et `distance_point_droite`, écrire une fonction `distance(p,a,b)` qui renvoie la distance entre le point P et la droite (AB) si les points A et B sont distincts et la distance AP sinon.

Dans la suite, on dira que la fonction `distance` calcule la distance entre le point P et les points A et B, éventuellement confondus.

3. On a besoin d'une fonction `le_plus_loin(ligne)` qui prend en argument une liste de tuples représentant les coordonnées des points d'une ligne polygonale. Cette fonction doit renvoyer un tuple composé de :
   - l'indice du point de coordonnées `p` de la ligne polygonale d'extrémités `deb` et `fin`, pour lequel la distance `distance(p, deb, fin)` est la plus grande ;
   - la valeur correspondante de cette distance.

   On fournit le code incomplet suivant :

```
def le_plus_loin(ligne):
    n = len(ligne)
    deb = ligne[0]
    fin = ligne[n-1]
    dmax = 0
    indice_max = 0
    for idx in range(1, n-1):
        p = ...
        d = distance(p, deb, fin)
        if ...:
            ...
            ...
    return ...
```

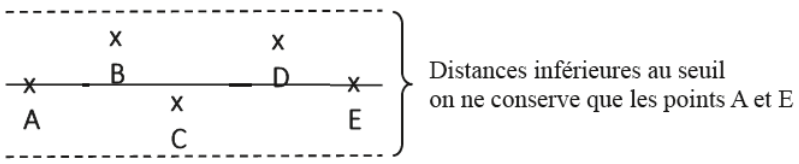
   Recopier et compléter le code de cette fonction.

4. Écrire une fonction `extrait(tab, i, j)` qui renvoie la copie du tableau `tab` des cases d'indice `i` inclus à `j` inclus pour `0 ≤ i ≤ j < len(tab)`. L'appel `extrait([7, 4, 9, 12], 2, 3)` renverra ainsi `[9, 12]`.

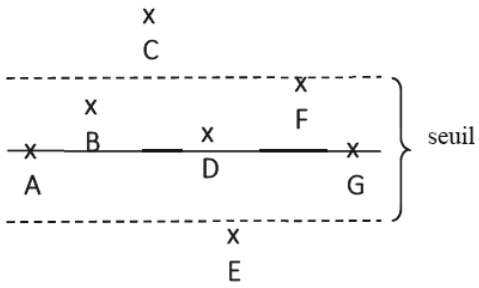
L'algorithme de Douglas-Peuker repose sur une stratégie de type « diviser pour régner ». Pour éliminer des sommets « proches de l'alignement », un seuil est fixé.

Étant donnée une ligne polygonale, le principe de l'algorithme est le suivant :

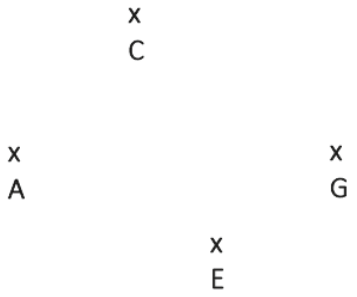
- si la ligne ne contient qu'un ou deux sommets, l'algorithme se termine ;
- sinon, on considère la droite formée par les extrémités de la ligne (son premier et dernier sommet), et on sélectionne le point le plus éloigné de cette droite (dans le cas où les extrémités sont confondues, on sélectionne le point le plus éloigné de celles-ci) :
  - si la distance entre le point sélectionné et la droite (ou les extrémités lorsqu'elles sont confondues) est inférieure au seuil fixé, on ne conserve que les extrémités de la ligne polygonale ;



- sinon, on applique l'algorithme de manière récursive aux deux parties de la ligne polygonale formées de la séquence formée du premier sommet jusqu'au sommet sélectionné d'une part et de la séquence formée du sommet sélectionné jusqu'au dernier sommet d'autre part. L'algorithme renvoie alors la concaténation des séquences simplifiées ainsi obtenues.



L'algorithme appelé sur la ligne polygonale [A,B,C,D,E,F,G] ci-dessus, va récursivement être appelé sur les lignes polygonales [A,B,C] et [C,D,E,F,G].
La ligne polygonale que l'on obtiendra à la fin de l'algorithme sera [A,C,E,G].



5. L'algorithme de Douglas-Peuker est implémenté par la fonction `simplifie` ci-dessous, qui prend en argument la ligne polygonale et le seuil choisi.

```
def simplifie(ligne, seuil):
    n = len(ligne)
    if n <= 2:
        return ...
    else:
        indice_max, dmax = le_plus_loin(ligne)

        if dmax <= seuil:
                return ...
        else:
                #bloc à écrire
```

Recopier et compléter le code de cette fonction.

Annexe (Exercice 1)

(à ne pas rendre avec la copie)

- **Types de données**

CHAR(t) VARCHAR(t) TEXT	Texte fixe de t caractères. Texte de t caractères variables. Texte de 65 535 caractères max.
INT	*Nombre entier de* -2^{31} à $2^{31}-1$ (signé) ou de 0 à $2^{32}-1$ (non signé)
FLOAT	Réel à virgule flottante
DATE DATETIME	Date format AAAA-MM-JJ Date et heure format AAAA-MM-JJHH:MI:SS

- **Quelques exemples de syntaxe SQL :**
  - Insérer des enregistrements :

```
INSERT INTO Table (attribut1, attribut2) VALUES(valeur1 , valeur2)
```

  - Modifier des enregistrements :

```
UPDATE Table SET attribut1=valeur1, attribut2=valeur2 WHERE Selecteur
```

  - Supprimer des enregistrements :

```
DELETE FROM Table WHERE Selecteur
```

  - Sélectionner des enregistrements :

```
SELECT attributs FROM Table WHERE Selecteur
```

  - Effectuer une jointure :

```
SELECT attributs FROM TableA JOIN TableB ON TableA.cle1=TableB.cle2 WHERE Selecteur
```