BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

Décembre 2023 – 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

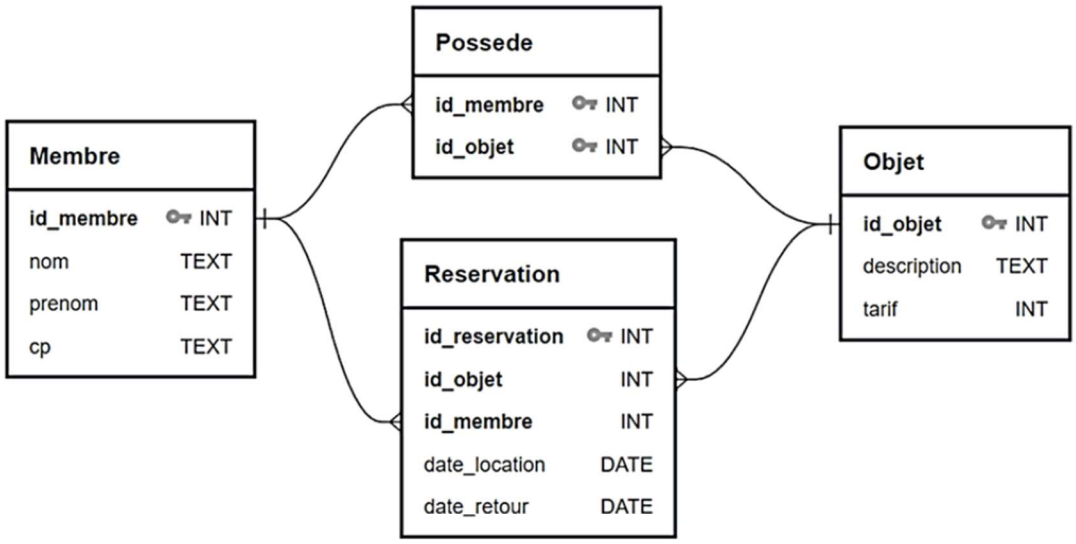
Dès que ce sujet vous est remis, assurez-vous qu'il est complet.
Ce sujet comporte 8 pages numérotées de 1 à 8

Exercice 1 : *(7 points)*

L'énoncé de cet exercice peut utiliser les mots du langage SQL suivants :
`SELECT, FROM, WHERE, JOIN ON, INSERT INTO, VALUES, UPDATE, SET, DELETE, COUNT, AND, OR`

Un site permet à ses membres de proposer à la location du matériel et de louer du matériel. Ceci permet de mutualiser du matériel entre membres et au propriétaire de rentabiliser cet achat. Le temps d'utilisation du matériel s'en trouve ainsi augmenté et le nombre d'appareils diminué.

Le modèle relationnel est donné par le schéma ci-dessous.
La table `Membre` contient les informations de chaque utilisateur du site (nom, prénom et code postal). La table `Objet` décrit le type d'objet à la location ainsi que son tarif de location journalier.
La table `Reservation` répertorie toutes les réservations effectuées par les membres du site avec notamment leur date de début et de fin de location. La table `Possede` permet de lier les tables `Membre` et `Objet`.



Conventions utilisées pour le schéma :

- Les clés primaires et étrangères sont mises en gras ;
- un symbole identifie une clé primaire ;
- un symbole entre deux attributs indique qu'ils doivent partager les mêmes valeurs et qu'ils sont reliés de la manière suivante : le côté + indique la clé primaire et le côté < indique la clé étrangère.

On donne ci-dessous le contenu de ces tables à un instant donné :

Membre

id_membre	nom	prenom	cp
1	"Ali"	"Mohamed"	"69110"
2	"Alonso"	"Fernando"	"69005"
3	"Dupont"	"Antoine"	"69003"
4	"Ferrand"	"Pauline"	"69160"
5	"Kane"	"Harry"	"69003"

Possede

id_membre	id_objet
1	4
1	6
2	4
3	3
3	5
4	1
4	2

Objet

id_objet	description	tarif
1	"Nettoyeur haute pression"	20
2	"Taille-haie"	15
3	"Perforatrice"	15
4	"Appareil à raclette"	10
5	"Scie circulaire"	15
6	"Appareil à gaufre"	10

Reservation

id_reservation	id_objet	id_membre	date_location	date_retour
1	4	5	2022-02-18	2022-02-19
2	1	2	2022-05-05	2022-05-06
3	3	1	2022-07-10	2022-07-12
4	3	1	2022-08-12	2022-08-14
5	2	2	2022-10-20	2022-10-22
6	2	2	2022-10-20	2022-10-22

1. Dans cette partie, on ne demande pas de requête SQL. En étudiant le contenu des tables ci-dessus :

   a. Indiquer quels sont les prénoms et noms du ou des membres du site qui proposent la location d'un appareil à raclette ;

   b. Donner le prénom et le nom du membre qui ne propose pas d'objet à la location.

2.

   a. Donner le résultat de la requête suivante :

```
SELECT nom, prenom FROM Membre WHERE cp = "69003";
```

   b. Écrire une requête permettant de connaître le tarif de location d'une scie circulaire.

   c. Écrire une requête permettant de modifier le tarif de location d'un nettoyeur à haute pression pour le passer à 15 € par jour au lieu de 20 € par jour.

   d. Écrire une requête SQL permettant d'ajouter Wendie Renard habitant à Villeurbanne (code postal 69100) dans la table `Membre`, avec un `id_membre` de 6.

3.

a. Expliquer la limitation importante d'utilisation du service offert par le site si l'on utilisait le couple de clés étrangères `(id_objet, id_membre)` en tant que clé primaire de la relation `Reservation`.

b. Mohamed Ali décide de ne plus être membre du site. Il faut donc le supprimer de la table `Membre` à l'aide de la requête :

```
DELETE FROM Membre
WHERE nom = "Ali" AND prenom ="Mohamed";
```

Expliquer pourquoi cette requête produit une erreur.

c. Proposer une suite de requêtes utilisant le mot clé **DELETE**, précédant la requête ci-dessus pour supprimer correctement Mohamed Ali, dont l'`id_membre` est 1, de la base de données.

Rappel : la relation `Objet` décrit le type d'objet à la location. Il n'y a pas d'objet à supprimer dans cette table lors du départ d'un membre.

4. Dans cette partie, les requêtes utilisent des jointures entre tables. On supposera donc les numéros `id_membre` et `id_objet` non connus.

a. Écrire une requête permettant de compter le nombre de réservations réalisées par Fernando Alonso.

b. Écrire une requête permettant de connaître les noms et prénoms des membres possédant un appareil à raclette.

Exercice 2 : *(6 points)*

Cet exercice porte sur les langages et la programmation (récursivité).

1. Voici une fonction codée en Python :

```
def f(n):
    if n == 0:
        print("Partez!")
    else:
        print(n)
        f(n-1)
```

a. Qu'affiche la commande `f(5)` ?

b. Pourquoi dit-on de cette fonction qu'elle est récursive ?

2. On rappelle qu'en python l'opérateur **+** a le comportement suivant sur les chaînes de caractères :

```
>>> S = 'a'+'bc'
>>> S
'abc'
```

Et le comportement suivant sur les listes :

```
>>> L = ['a'] + ['b', 'c']
>>> L
['a', 'b', 'c']
```

On a besoin pour les questions suivantes de pouvoir ajouter une chaîne de caractères `s` en préfixe à chaque chaîne de caractères de la liste `liste`. On appellera cette fonction `ajouter`.

Par exemple, `ajouter("a",["b","c"])` doit retourner `["ab","ac"]`.

a. Recopiez le code suivant et complétez `.....` sur votre copie :

```
def ajouter(s, liste):
    res = []
    for m in liste:
        res.....
    return res
```

b. Que renvoie la commande `ajouter("b", ["a","b","c"])` ?

c. Que renvoie la commande `ajouter("a", [""])` ?

3. On s'intéresse ici à la fonction suivante écrite en Python où `s` est une chaîne de caractères et `n` un entier naturel.

```
def produit(s, n):
    if n == 0:
        return [""]
    else:
        res = []
        for i in range(len(s)):
            res = res + ajouter(s[i], produit(s, n-1))
        return res
```

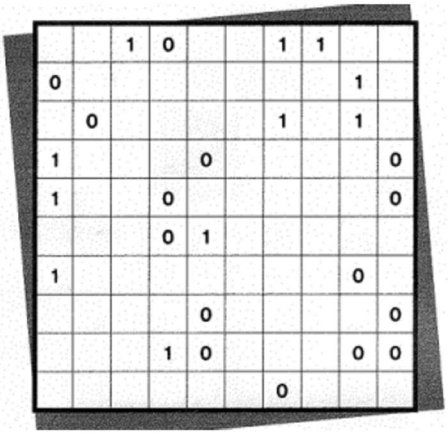
a. Que renvoie la commande `produit("ab", 0)` ? Le résultat est-il une liste vide ?

b. Que renvoie la commande `produit("ab", 1)` ?

c. Que renvoie la commande `produit("ab", 2)` ?

Exercice 3 : *(7 points)*

Cet exercice porte sur les algorithmes et la programmation Python.



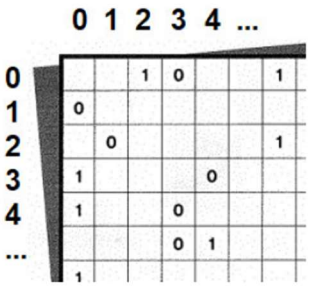
Dans le jeu du **TAKAZU**, on dispose d'une grille de 10 lignes et 10 colonnes contenant des zéros et des uns. L'objectif est de compléter les cases blanches en respectant les règles ci-dessous :

- **[REGLE1]** : chaque ligne et chaque colonne doivent contenir autant de 0 que de 1.
- **[REGLE2]** : les lignes ou colonnes identiques sont interdites.
- **[REGLE3]** : il ne doit pas y avoir plus de deux 0 ou plus de deux 1 placés à la suite, ni dans le sens vertical, ni dans le sens horizontal.

Dans cet exercice, on suppose que les valeurs de chaque ligne sont stockées dans une liste en Python, et toutes les lignes sont à leur tour placées dans une liste **globale** notée `grille`.

Ainsi, dans la situation représentée ci-contre, `grille[0][2]` vaut 1 et `grille[0][3]` vaut 0.

On décide aussi de coder par -1 toutes les cases blanches, c'est-à-dire celles dont on ne connait pas encore la valeur. Dans notre exemple, au départ `grille[0][1]` vaut -1.



1. Ecrire une fonction `autre(x)` qui renvoie 1 si `x` vaut 0 et 0 si `x` vaut 1.

2. D'après la première règle, si une ligne contient 5 zéros, les cases blanches de cette ligne sont forcément des uns. De même si une ligne contient 5 uns, les cases blanches sont forcément des zéros.

a. Ecrire le code d'une fonction `nbValeurs(li, v)` dont les paramètres sont `li`, le numéro de la ligne de la grille et `v` la valeur que l'on souhaite compter et qui retourne la nombre de fois où la valeur `v` est présente sur la ligne `li`.

Sur l'exemple

```
>>>nbValeurs(0,1)
3
```

b. En déduire une fonction `regle1(li)` dont le paramètre `li` indique le numéro de la ligne à remplir et qui modifie l'objet `grille` en remplissant de 0 ou de 1 à partir de la première règle. Par exemple si la 6ème ligne est

		0		0	0	0		1	0

après l'appel `regle1(6)`, la ligne devient

1	1	0	1	0	0	0	1	1	0

. S'il n'y a pas assez de 0 ou de 1, la fonction ne modifie rien.

3. L'extrait de code ci-dessous effectue la recherche de deux cases identiques séparées par une blanche. Dans ce cas, on peut déterminer la valeur de la case blanche.

Par exemple, si la ligne n°4 est | | | 0 | | 0 | 1 | 0 | | 1 | | |, il y a deux zéros séparés par une case blanche donc la case blanche contient nécessairement un 1 sinon la troisième règle n'est pas respectée. On complètera donc en | | | 0 | 1 | 0 | 1 | 0 | | 1 | | |

Recopier et compléter les [.....] de la fonction `regle3(li)` dont le paramètre `li` indique le numéro de la ligne étudiée et qui modifie l'objet grille en utilisant la remarque précédente.

```
def regle3(li)
   for col in range(…):
     if grille[li][col] == grille[li][col+2] and [.....]:
              grille[.....] = autre[.....]
```

4. On suppose pour cette question que toutes les cases blanches ont été complétées. On veut déterminer si toutes les lignes sont bien différentes pour respecter la seconde règle. Chaque ligne constituée de 0 et de 1 peut être considérée comme un nombre écrit en base 2.

Ecrire le code d'une fonction `convert(L)` dont le paramètre est une liste `L` de 10 bits (0 ou 1) et renvoie la valeur décimale correspondant à la ligne `L`.

Par exemple `convert([0,0,0,0,0,1,1,1,1,1])` doit renvoyer `31`

5. On considère que l'on a stocké les valeurs obtenues à l'aide de la fonction `convert` dans une nouvelle liste `V`. On souhaite s'assurer qu'il n'y a pas de doublon.

Proposer en langage naturel l'algorithme d'une fonction dont le paramètre est une liste d'entiers et qui renvoie un booléen (VRAI ou FAUX) indiquant si cette liste possède des doublons.