EXERCICE 2 (6 points)

Cet exercice porte sur la programmation Python, la programmation orientée objet, les bases de données relationnelles et les requêtes SQL.

Cet exercice est composé de deux parties indépendantes.

Partie A

Dans cette partie, nous allons utiliser une base de données sur les champignons. Le tableau ci-dessous nous donne un extrait du résultat obtenu, après avoir effectué la requête suivante :

```
SELECT * FROM champignon;
```

id	nom	id_ordre	lamelle	couleur	chapeau_min	chapeau_max	pied_min	pied_max
1	champignon de Paris	1	oui	blanc	4	10	2	5
2	champignon noir	2	non	noir	2	10	0	0
3	coprin chevelu	1	oui	blanc	5	20	10	40
4	bolet à pied rouge	3	non	jaune	7	19	5	15
5	amanite des Césars	4	oui	orange	8	20	8	15

Les attributs `nom`, `lamelle` et couleur sont des chaines de caractères. La taille du chapeau et la longueur du pied des champignons sont des nombres exprimés en centimètres.

1. Mathilde vient de trouver un champignon avec des lamelles et de couleur orange. Écrire la requête qu'elle doit écrire pour avoir la liste des noms des champignons possibles pour sa trouvaille.

2. Par ailleurs, Romain a trouvé un champignon sans pied et un chapeau de 15 cm. Écrire la requête qu'il doit écrire pour avoir la liste des noms des champignons possibles pour son spécimen.

Les champignons sont classés selon différents ordres selon la table `ordre` suivante où l'on trouve également la classe à laquelle appartient chaque ordre. Voici le contenu de la table `ordre`.

table `ordre`		
`id`	`nom`	`classe`
1	agaricales	agaricomycètes
2	trémellales	phragmabasidiomycètes
3	bolétales	agaricomycètes
4	amanitales	agaricomycètes
5	cantharellales	agaricomycètes
6	polyporales	basidiomycètes
7	clavariales	homobasidiomycètes
8	tricholomatales	agaricomycètes

3. Donner la clé étrangère de la table `champignon` faisant référence à la clé primaire `id` de la table `ordre`.

4. Écrire la requête permettant d'obtenir la liste des noms des champignons appartenant à la classe des `agaricomycètes`.

Le champignon nommé amanite solitaire est un champignon blanc qui appartient à l'ordre des amanitales. Il possède des lamelles. Son chapeau mesure entre 6 et 20 cm et son pied entre 4 et 10 cm.

5. Écrire une requête permettant d'ajouter ce champignon avec l'`id` 56 dans la table `champignon`.

On décide d'ajouter les différentes toxicités des champignons en ajoutant un attribut `id_toxicite` dans la relation `champignon` et une table `toxicite`. Voici le contenu de la table `toxicite` :

`id_tox`	`type`	`effets`
1	très toxique	Entrainant la mort
2	toxique	Entrainant des problèmes digestifs ou nerveux
3	à rejeter	Champignons suspects ou ayant un mauvais gout
4	comestible	Champignons pouvant être consommés

6. Écrire le schéma relationnel de la base de données contenant ces trois tables en soulignant les clés primaires et en faisant précéder les clés étrangères du caractère #.

Une erreur s'est glissée dans cette table : le champignon de nom amanite citrine est très toxique car son ingestion peut entrainer la mort.

7. Écrire la requête permettant de corriger cette erreur dans la table `champignon`.

8. Écrire une requête permettant d'obtenir les noms des champignons de l'ordre des `amanitales` de toxicité `très toxique`.

Partie B

Quentin fait quelques recherches afin de trouver les recettes possibles avec des champignons selon les saisons. Il trouve notamment les informations suivantes pour le lactaire délicieux :

Lactaire délicieux :

- localisation : sous les pins
- saison : été
- recette : lactaires grillés à l'huile d'olive
- cuisson : 12 minutes à feu moyen

Pour chaque champignon, il souhaite stocker les informations dans un objet de la classe `Champignon` définie ci-dessous :

```
1 class Champignon():
2     def __init__(self, nom, lieu, saison, recette, cuisson):
3         self.nom = nom
4         self.localisation = lieu
5         self.saison = saison
6         self.recette = recette
7         self.cuisson = cuisson
```

Donc l'instruction permettant d'instancier l'objet `champignon1` du champignon `lactaire délicieux` est la suivante :

```
champignon1 = Champignon('Lactaire délicieux', 'Sous les pins',
'été', 'Lactaires grillés à l\'huile d\'olive', '12 minutes à feu
moyen')
```

Quentin décide de stocker l'ensemble des instances de classe Champignon dans une variable liste_champi de type list.

9. Recopier et compléter le programme ci-dessous permettant d'afficher le(s) nom(s) des champignons que l'on trouve lors de la saison `été`.

```
for e in liste_champi:
    if ..... == 'été':
        print(...)
```

La plaque de cuisson de Quentin est défectueuse. Il ne peut cuire des aliments qu'à feu moyen. Il écrit donc rapidement le programme suivant afin de savoir s'il peut cuire ses champignons à feu moyen.

```
1 for c in liste_champi:
2     if c.nom == 'Lactaire délicieux':
3         return c.cuisson == 'feu moyen'
```

Ce programme renvoie `False`, alors que le lactaire délicieux se cuisine bien à feu moyen.

10. Expliquer pourquoi il n'obtient pas le résultat attendu alors que Quentin n'a fait aucune erreur dans sa liste de champignons.

Malgré ce problème, Quentin ne souhaite pas modifier son travail et téléphone à son ami Iris qui lui fournit une fonction `recherche_textuelle` qui prend en paramètre deux chaines de caractères `texte` et `mot` et qui retourne `True` si `mot` est un mot appartenant à `texte` et `False` sinon.

```
>>> recherche_textuelle('dans une forêt', 'forêt')
True
>>> recherche_textuelle('près de la mer', 'mare')
False
```

11. Corriger le programme de la question précédente en utilisant la fonction `recherche_textuelle`.