



Concours Biologie et Géologie Epreuve d'Informatique

Date : Mercredi 8 Juin 2022 Heure : 8 H Durée : 2 H Nombre de pages : 9
Barème : PROBLEME 1 : 7 points
PROBLEME 2 : 13 points

DOCUMENTS NON AUTORISÉS
L'USAGE DES CALCULATRICES EST INTERDIT
II FAUT RESPECTER IMPERATIVEMENT LES NOTATIONS DE L'ENONCE
VOUS POUVEZ EVENTUELLEMENT UTILISER LES FONCTIONS PYTHON
DECRIRES À L'ANNEXE (PAGE 9/9)

Le sujet comporte deux problèmes indépendants portant sur la même thématique présentée dans la description générale qui suit :

Description générale :

Un Quiz est un questionnaire permettant de tester une ou plusieurs compétence(s) générale(s) ou spécifique(s). Un Quiz est utilisé dans l'enseignement pour des fins d'évaluation formative et/ou sommative. Il est aussi utilisé dans les examens de certification. Différents types de questions existent, dont les plus utilisées sont : les Questions à Choix Multiples (QCM) auxquelles le candidat doit répondre en sélectionnant au moins une parmi une liste de propositions de réponses possibles associées à la question. Une ou plusieurs de ces proposition(s) est/sont correcte(s). Les autres sont des propositions erronées dites distracteurs.

La Figure 1, illustre à travers un exemple, les différentes parties d'une question QCM qui vise la compétence « Bases de Données Relationnelles ».

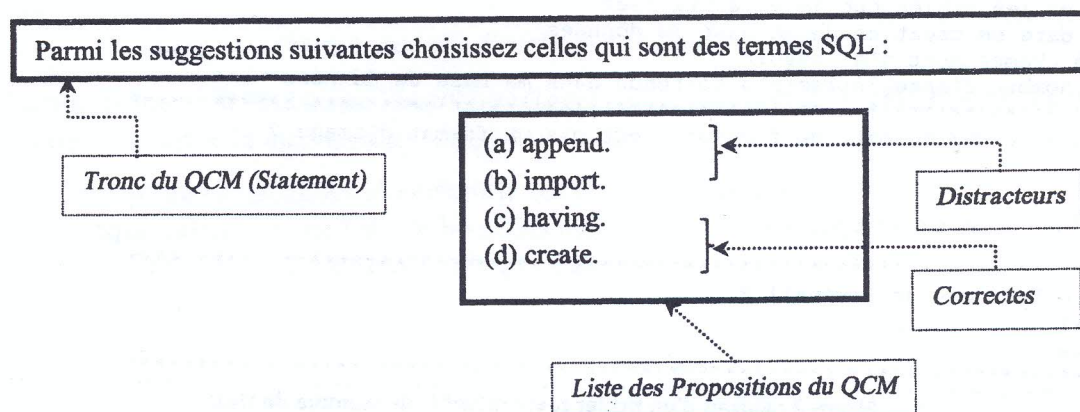


Figure 1 : Différentes parties d'une question QCM

Toute question appartenant à un Quiz est notée sur 1 point.
 Au cours du passage du Quiz un candidat doit distinguer la ou les proposition(s) qu'il juge exacte(s) en les choisissant, de celles qu'il juge distracteurs.
 À la fin du passage d'un Quiz, un score est calculé pour décider si un candidat est digne de certification.
 Le score est la somme des notes obtenues dans toutes les questions du Quiz.
 Le tableau suivant présente le barème adopté pour la notation d'une question QCM.

Tableau 1 : Barème de notation d'une question QCM

Notation	Par réponse correcte	Par réponse incorrecte
k = nombre total de propositions	$+\frac{1}{c}$	$-\frac{1}{k-c}$
c = nombre de propositions correctes		

Pour la question de la Figure 1, la notation est calculée comme suit :

- c = nombre de propositions correctes = 2
 - k = nombre total de propositions = 4.
- ⇒ On accorde $+\frac{1}{c}=+0.5$ par proposition correcte choisie et $-\frac{1}{k-c}=-0.5$ par proposition incorrecte (distracteur) choisie.

Par exemple, pour deux candidats et qui ont émis respectivement les deux réponses "a c" et "a b d" auront dans l'ordre les scores $0 = (-0.5 + 0.5)$ et $-0.5 = (-0.5 - 0.5 + 0.5)$

PROBLEME 1

Le but de ce problème est d'implémenter en Python des fonctions permettant d'extraire, de structurer les données d'un Quiz à partir d'un fichier en format texte, de récupérer les réponses d'un candidat et de calculer son score.

Pour ce faire on dispose du fichier nommé "Quiz.txt" formaté comme suit (voir Figure 2) :

- Chaque question est sur une seule ligne ;
- Chaque bonne proposition est sur une seule ligne qui commence par le caractère '+' ;
- Chaque mauvaise proposition (distracteur) est sur une seule ligne qui commence par le caractère '-' ;
- Les questions sont séparées par une ligne à base d'une succession du caractère '*' ;

```

Pour créer une requête dans une base de données relationnelle j'ai besoin de savoir :
+la ou les tables que je vais utiliser
-la date de création de la base de données
+les champs dont j'ai besoin
-le nombre d'enregistrements contenus dans ma base de données
*****
Lequel de ces formats de fichier n'est pas un format d'image ?
-GIF
-PNG
+ODT
-BMP
*****
Un routeur est un firewall ?
+Vrai
-Faux
*****
    
```

Figure 2 : Extrait d'un fichier texte relatif à un exemple de Quiz

À partir du fichier "Quiz.txt", on construit les dictionnaires nommées respectivement **Ques**, **Prop** et **Rep** tels que:

Ques est un dictionnaire contenant toutes les questions du Quiz où :

- Chaque clé est un entier identifiant le numéro de la question, généré selon l'ordre de présence des questions dans le fichier.
- Chaque valeur est une chaîne contenant le tronc (énoncé) de la question

Prop est un dictionnaire contenant les propositions de toutes les questions du Quiz où :

- Chaque clé est un entier identifiant le numéro de la question.
- Chaque valeur est une liste de propositions. Une proposition est une chaîne de caractères.

Rep est un dictionnaire contenant toutes les réponses aux questions du Quiz où :

- Chaque clé est un entier identifiant le numéro de la question.
- Chaque valeur est une liste de booléens, True si la réponse est correcte, False sinon.

Travail demandé :

Dans la suite les fonctions demandées seront écrites en langage Python

1. Écrire la fonction **ExtraireQuiz** qui prend en entrée le nom du fichier texte **NomFich**, retourne un tuple contenant les trois dictionnaires : **Ques**, **Prop** et **Rep**.

Le résultat de l'extraction des données à partir du fichier texte est illustré dans la Figure 3.

```
Ques= {1: "Pour créer une requête dans une base de données relationnelle j'ai besoin de savoir :", 2: "Lequel de ces formats de fichier n'est pas un format d'image ? ", 3: " Un routeur est un firewall ?"}

Prop= {1: ["la ou les tables que je vais utiliser", "la date de création de la base de données", "les champs dont j'ai besoin ", "le nombre d'enregistrements contenus dans ma base de données"], 2: ["GIF", "PNG", "ODT", "BMP"], 3: ["Vrai", "Faux"]}

Reponses={1: [True, False, True, False], 2: [False, False, True, False], 3: [True, False]}
```

Figure 3 : Résultat de l'extraction des données relatives à la Figure 2

2. Écrire la fonction **AfficheQuestion** qui prend en entrée le dictionnaire **Ques**, le dictionnaire **Prop** et un entier **num** représentant un numéro de question. La fonction affiche l'énoncé de la question ainsi que ses différentes propositions numérotées alphabétiquement à partir de la lettre (a) comme illustré dans la Figure 4.

L'appel de **AfficheQuestion(Ques, Prop, 2)** donne le résultat suivant :

```
2) Lequel de ces formats de fichier n'est pas un format d'image?
(a) GIF
(b) PNG
(c) ODT
(d) BMP
```

Figure 4 : Exemple d'affichage d'une question QCM

3. Écrire la fonction **InitialiserDictCandidat** qui prend en entrée le dictionnaire **Prop**, retourne le dictionnaire **dictCand** où :
 - Chaque clé est un entier identifiant le numéro de la question.
 - Chaque valeur est une liste de booléens initialisés à False. cette liste a la même taille que la liste des propositions de chaque question.

4. Écrire la fonction **LectureRepQCM** qui prend en entrée le dictionnaire **dictCand** et un entier **num**, permettant de :
 - Saisir une chaîne contenant la/les lettre(s) alphabétique(s) associée(s) au(x) choix du candidat (séparées par des espaces). On ne demande pas le contrôle de saisie des lettres correspondantes aux choix.
 - Demander au candidat de confirmer son choix en affichant les propositions choisies accompagnées du message « Voulez-vous confirmer votre réponse O/ N ? ».
 - Mettre à jour le dictionnaire **dictCand** si le candidat confirme par « O ».
 - Répéter le processus si le candidat répond par « N ».

L'appel de **LectureRepQCM (dictCand, 2)** donne le résultat suivant :

```

2) Lequel de ces formats de fichier n'est pas un format ?
(a) : GIF
(b) : PNG
(c) : ODT
(d) : BMP
Donner la/les lettre(s) alphabétique(s) associée(s) au(x) choix du votre réponse
(séparés par des espaces) : a c
Voulez-vous confirmer votre réponse O/ N ? N

Donner la/les lettre(s) alphabétique(s) associée(s) au(x) choix du votre réponse
(séparés par des espaces) : a b
Voulez-vous confirmer votre réponse O/ N ? O
  
```

Figure 5 : Exemple de saisie de réponses.

5. Écrire la fonction **NoterRepQCM** qui prend en entrée le dictionnaire **dictCand**, le dictionnaire **Rep** et un entier **num** représentant un numéro de question. La fonction calcule et retourne la note obtenue selon le barème décrit dans le Tableau 1.
6. Écrire le script Python permettant de :
 - Construire les dictionnaires **Ques**, **Prop** et **Rep** ;
 - Initialiser le dictionnaire **dictCand** ;
 - Pour chaque question du dictionnaire **Ques** :
 - Afficher la question suivie de ses propositions ;
 - Saisir la réponse du candidat ;
 - Mettre à jour **dictCand** selon ses choix ;
 - Calculer et afficher le score obtenu par le candidat sous forme de pourcentage par rapport au nombre total de questions.

PROBLEME 2

Un cabinet de formation dispose d'une base de données pour la gestion de son référentiel de questions d'évaluation et de certification.

Il y a lieu de considérer les points suivants :

- Un candidat obtient un voucher pour passer un Quiz (examen de certification).
- Un candidat peut acheter plusieurs vouchers pour le passage de divers examens de certification.
- Toutes les questions sont de type QCM y compris les questions de type Vrai/Faux.
- Un Quiz passé par un candidat suite à l'achat d'un voucher, est appelé tentative.
- Pour être évaluée, une tentative de candidat doit être validée et soumise au cours d'un délai de temps publié pour l'examen de certification.
- Dépassant la durée prévue pour un examen de certification sans soumission, le candidat est considéré défaillant.

- Pour répondre aux questions du Quiz, le candidat doit choisir les réponses qu'il juge correctes.
- Toute question appartenant à un Quiz est notée sur 1 point.
- Le calcul de note d'un candidat pour toute question QCM du quiz est fait sur la base du barème expliqué dans le Tableau 1.
- Le score d'un candidat dans une tentative est un pourcentage calculé sur la base de la somme des notes obtenues pour toutes les questions du Quiz et divisé par le nombre des questions.
- Une question QCM peut avoir des propositions de réponses toutes entièrement correctes mais jamais toutes fausses.
- Une question QCM de type Vrai/faux, supporte uniquement deux propositions de réponses, exclusivement une correcte, et une fausse.

Le schéma relationnel de la base de données est défini comme suit :

▪ **Quiz** (idQuiz, description, duree, dateCreation, dateFin)

La table **Quiz** enregistre les informations relatives à un Quiz.

- idQuiz : identifiant du Quiz, un nombre de type entier, clé primaire ;
- description : décrit l'intitulé de l'examen de certification, de type chaîne ;
- duree : durée officielle du Quiz, de type entier exprimée en minutes ;
- dateCreation : date de création relative à la mise en vigueur du Quiz, de type date ;
- dateFin : date fin de mise en vigueur du Quiz, de type date ;

▪ **Question** (idQs, tronc, dateCr)

La table **Question** enregistre les informations générales relatives à une question.

- idQs : identifiant d'une question, clé primaire, de type entier;
- tronc : énoncé de la question, de type chaîne;
- dateCr : date de création de la question, de type date ;

▪ **Proposition** (#idQs, idProp, textProp, veracite)

La table **Proposition** enregistre les propositions de réponses relatives à une question (les réponses correctes et les distracteurs).

- idQs : identifiant d'une question, clé étrangère qui fait référence à la table **Question**;
- idProp : identifiant d'une réponse proposée relativement à la question identifiée par idQs, de type entier;
- textProp : réponse proposée, de type chaîne;
- veracite : indique l'état de la réponse proposée, de type entier, égale à 1 si la réponse proposée est vraie, 0 sinon;

NB : idQs et idProp ensembles forment la clé primaire de la table **Proposition**.

▪ **DetailQuiz** (#idQuiz, #idQs)

La table **DetailQuiz** enregistre les questions relatives à un Quiz.

- idQuiz : identifiant d'un Quiz, clé étrangère qui fait référence à la table **Quiz** ;
- idQs : identifiant d'une question, clé étrangère qui fait référence à la table **Question** ;

NB : idQuiz et idQs ensembles forment la clé primaire de la table **DetailQuiz**.

▪ **Candidat** (idCand, nom, prenom, email, pwd)

La table **Candidat** enregistre les informations relatives aux candidats qui passent des examens de certification.

- idCand : identifiant du candidat, clé primaire, de type chaîne ;
- nom : nom du candidat, de type chaîne ;
- prenom : prénom du candidat, de type chaîne ;
- email : adresse mail du candidat, de type chaîne ;
- pwd : mot de passe du candidat, de type chaîne ;

▪ **Tentative** (voucher, #idCand, #idQuiz, datehDeb, datehFin, scoreG)

La table **Tentative** enregistre les informations relatives à un passage d'un examen de certification par un candidat suite à l'achat d'un voucher.

- voucher : identifiant délivré à tout candidat ayant payé des frais et désirant passer une certification, clé primaire, de type chaîne ;
- idCand : identifiant d'un candidat, clé étrangère qui fait référence à la table **Candidat** ;
- idQuiz : identifiant d'un Quiz, clé étrangère qui fait référence à la table **Quiz** ;
- datehDeb : date et heure début de passage de l'examen de certification, de type datetime ayant le format 'AAAA-MM-JJ hh:mm:ss' ;
- datehFin : date et heure fin de passage de l'examen de certification, de type datetime ayant le format 'AAAA-MM-JJ hh:mm:ss' ;
- scoreG : score obtenu par le candidat sur la base de ses réponses aux questions proposées, de type réel ;

▪ **ReponseCandidat**(#voucher, #idQs, #idProp)

La table **ReponseCandidat** enregistre les propositions choisies explicitement par le candidat.

- voucher : identifiant d'une tentative, clé étrangère qui fait référence à la table **Tentative** ;
- idQs : identifiant d'une question, clé étrangère qui fait référence à la table **Question** ;
- idProp : identifiant d'une proposition, clé étrangère qui fait référence à la table **Proposition** ;

Travail demandé :

Partie 1

Pour chacune des requêtes algébriques suivantes, décrire le résultat attendu.

1.

$$\prod_{idQuiz} (Quiz) - \prod_{idQuiz} (Tentative)$$

2.

$$\prod_{trunc, textProp} \left(\sigma_{veracite=1} \left(Question \bowtie_{idQs} Proposition \right) \right)$$

Partie 2

Exprimer en SQL les requêtes suivantes permettant de :

1. Créer la table **Proposition** en respectant les contraintes d'intégrité spécifiées dans le schéma relationnel.

Dans la suite on suppose que les tables sont créées et remplies.

2. Ajouter la question d'identifiant 17546 au Quiz d'identifiant 228.
3. Supprimer les questions dont les dates de création sont antérieures à l'année 2012.
4. Déterminer les noms des candidats qui comportent les lettres "L" et "A".
5. Déterminer les troncs des questions et leurs propositions de réponses relatives au Quiz ayant l'identifiant 175.
6. Déterminer les informations des Quiz qui ne seront plus en vigueur à partir d'aujourd'hui sachant qu'une date peut être comparée à la valeur de retour de la fonction `current_date` qui représente la date courante.
7. Déterminer les identifiants, troncs et propositions des questions qui n'ont jamais été assignées dans des Quiz.
8. Déterminer les identifiants des questions QCM de type Vrai/Faux.
9. Déterminer les troncs des questions QCM qui proposent plus que 4 réponses possibles.
10. Déterminer les identifiants des questions QCM qui proposent de 3 à 5 réponses, toutes correctes.
11. Déterminer les informations des tentatives des candidats défaillants qui ont dépassé le délai réglementaire prévu pour le passage de tout Quiz de certification. La fonction `JulianDay(datetime)` permet de calculer l'intervalle de temps entre deux dates et heures précises. Les exemples suivants expliquent l'utilisation de la fonction `JulianDay`

Exemple 1 :

```
SELECT(julianday('2022-06-01 10:00:00')-julianday('2022-06-01 08:00:00')); donne 0.0833333330228925
```

Exemple 2 :

```
SELECT(julianday(2022-06-08 10:00:00')-julianday(2022-06-01 08:00:00')); donne 7.08333333302289
```

12. Déterminer les identifiants des Quiz qui ne proposent pas des questions QCM de type Vrai/Faux.

Partie 3

Dans la suite les fonctions demandées doivent être écrites en Python. On désigne par **Cur** le curseur d'exécution de requêtes.

1. Écrire la fonction **Score_Question** qui prend en paramètre **Cur**, l'identifiant **V** d'un voucher et l'identifiant **idQ** d'une question, calcule la note obtenue par le candidat.
2. Écrire la fonction **Score_Quiz** qui prend en paramètre **Cur**, l'identifiant **V** d'un voucher puis met à jour le score obtenu par le candidat.

3. Écrire la fonction **Stat_Certif** qui prend en paramètre **Cur** et renvoie une liste de tuples. Chaque tuple est constitué de cinq éléments : l'identifiant du Quiz et quatre indicateurs de performance (le score maximum, le score moyen, le score minimum et l'écart type).

La formule de l'écart type est :
$$\sigma = \sqrt{\frac{\sum_{i=1}^n |x_i - \mu|^2}{n}}$$

x_i : désigne le score d'un candidat pour un Quiz.

μ : désigne la moyenne des scores des candidats qui ont passé ce Quiz.

n : désigne l'effectif des candidats qui ont passé ce Quiz.

ANNEXE – Quelques Fonctions/Méthodes Python

Sans aucune obligation, les fonctions suivantes pourraient vous être utiles.

Opérations sur les fichiers

<code>f=open(nomF,m)</code>	Permet d'ouvrir le fichier <code>nomF</code> en mode <code>m</code> ou <code>m='r'</code> ou <code>'w'</code> .
<code>f.close()</code>	Permet de fermer un fichier.
<code>f.read()</code>	Permet de lire et retourner le contenu d'un fichier dans une chaîne de caractères.
<code>f.readline()</code>	Permet de lire et retourner la ligne courante d'un fichier dans une chaîne de caractères.
<code>f.readlines()</code>	Permet de lire et retourner le contenu de toutes les lignes d'un fichier dans une liste.

Opérations sur les itérables (str, tuple, list, dict, etc.)

<code>len(it)</code>	Retourne le nombre d'éléments de l'itérable <code>it</code> .
<code>range(d,f,p)</code>	Retourne la séquence des valeurs entières successives comprises entre <code>d</code> et <code>f</code> , <code>f</code> exclu, par <code>pas=p</code> .
<code>x in it</code>	Vérifie si <code>x</code> appartient à <code>it</code> .
<code>n*it</code>	Concatène <code>n</code> copies de <code>it</code> .
<code>ord(c)</code>	Retourne le code ASCII du caractère <code>c</code> . Par exemple, <code>ord('a')</code> renvoie l'entier 97.
<code>chr(code)</code>	Retourne le caractère correspondant au code ASCII.
<code>st.split(motif)</code>	Retourne une liste formée par des chaînes de caractères résultantes du découpage de la chaîne <code>st</code> autour de la chaîne <code>motif</code> .
<code>st.strip()</code>	Retourne une copie de <code>st</code> en supprimant à la fois les retours à la ligne de fin et de début.
<code>st.startswith(prefix)</code>	Retourne <code>True</code> si la chaîne <code>st</code> commence par la chaîne <code>prefix</code> .
<code>motif.join</code> (itérable de chaînes)	Retourne une chaîne de caractères résultante de la concaténation des éléments de l'itérable intercalés par le <code>motif</code> .
<code>motif.format(paramètres)</code>	Retourne une chaîne de caractères obtenue en substituant dans l'ordre chaque caractère <code>{}</code> dans <code>motif</code> par un objet dans <code>paramètres</code> .
<code>d.keys()</code>	Retourne un itérable formé par les clés du dictionnaire <code>d</code> .
<code>d.values()</code>	Retourne un itérable formé par les valeurs du dictionnaire <code>d</code> .
<code>d.items()</code>	Retourne un itérable de couples (<code>k,v</code>) ou <code>k</code> est une clé du dictionnaire <code>d</code> et <code>v</code> est la valeur associée.
<code>lst.count(val)</code>	Retourne le nombre d'occurrences de <code>val</code> dans la liste <code>lst</code> .
<code>lst.append(val)</code>	Ajoute <code>val</code> à la fin de la liste <code>lst</code> .