Amérique du Nord – 2025 – sujet2 - Correction

Exercice 3 (8 points)

PARTIE A – Bases de données

1. Expliquer le choix de `num_dossard` comme clé primaire pour la relation `coureur`.

Le champ `num_dossard` est une bonne clé primaire car il identifie de manière unique chaque coureur. Deux coureurs ne peuvent pas avoir le même numéro de dossard.

👉*Commentaire : Une clé primaire doit être unique et permettre d'identifier un seul enregistrement dans une table.*

---

2. Décrire ce que donne la requête SQL suivante.

```sql
SELECT nom, prenom
FROM coureur
ORDER BY nom;
```

Cette requête affiche les noms et prénoms de tous les coureurs, triés par ordre alphabétique des noms.

👉*Commentaire : `ORDER BY nom` effectue un tri croissant par défaut.*

---

3. Écrire une requête SQL permettant d'établir le nom et prénom de toutes les femmes inscrites à la compétition.

```sql
SELECT nom, prenom
FROM coureur
WHERE sexe = 'F';
```

👉*Commentaire : La condition `WHERE sexe = 'F'` permet de sélectionner uniquement les femmes.*

---

4. Écrire une requête SQL permettant de connaître le nombre total d'inscrits à la compétition.

```sql
SELECT COUNT(*)
FROM coureur;
```

👉*Commentaire : `COUNT(*)` compte le nombre total de lignes dans la table.*

---

5. Écrire une requête SQL permettant d'insérer ce participant dans la base.

```sql
INSERT INTO coureur(nom, prenom, annee, sexe, id_epreuve, temps)
VALUES ('REMY', 'Patrice', 1973, 'H', 1, 0);
```

👉*Commentaires :*

- *Le numéro de dossard n'est pas indiqué car il est incrémenté automatiquement.*
- *La partie en gris n'est pas necessaire si les valeurs sont insérées dans l'ordre des attributs de la table.*

---

6. Écrire une requête SQL qui permettra à l'organisateur de supprimer son inscription de la base de données.

```sql
DELETE FROM coureur
WHERE num_dossard = 137;
```

👉*Commentaire : La clause `WHERE` est indispensable pour éviter de supprimer toute la table.*

---

7. Écrire une requête SQL permettant de fournir la distance et l'horaire du départ du coureur 256.

```sql
SELECT distance, horaire
FROM coureur
JOIN epreuve ON coureur.id_epreuve = epreuve.id_epreuve
WHERE num_dossard = 256;
```

👉*Commentaire : La jointure permet de relier les informations du coureur à celles de l'épreuve.*

---

8. Écrire une requête SQL renvoyant le numéro de dossard, le nom, le prénom et le temps de course des participantes Master Femme du 10 km classées par temps croissant.

```sql
SELECT num_dossard, nom, prenom, temps
FROM coureur
JOIN epreuve ON coureur.id_epreuve = epreuve.id_epreuve
WHERE sexe = 'F'
AND annee < 1986
AND distance = 10
ORDER BY temps;
```

👉*Commentaire : Les coureuses « Master Femme » sont nées avant 1986. Le tri se fait ici du plus petit temps au plus grand.*

PARTIE B – Programmation Python

On rappelle le dictionnaire :

```python
dict_perf_5km = {
    2022 : [1020,1050,900,1000,1018,1040],
    2023 : [1010,1048,1100,1024,1080,1108],
    2024 : [1012,1010,1000,1036,1022,1098],
    2025 : [998,1028,1000,959,1002,980]
}
```

---

9. Donner la valeur de l'expression `dict_perf_5km[2025][2]`.

```
1000
```

👉*Commentaire : `dict_perf_5km[2025]` renvoie la liste associée à 2025, puis `[2]` récupère le troisième élément de cette liste.*

---

10. Écrire l'instruction permettant d'ajouter les résultats de 2026.

```
dict_perf_5km[2026] = [1004, 1016, 1000, 1140, 1023, 1024]
```

👉*Commentaire : Ajouter une nouvelle entrée dans un dictionnaire se fait avec une nouvelle clé et sa valeur associée.*

---

11. Écrire le code de la fonction `scratch`.

```python
def scratch(dico, annee):
    meilleur = dico[annee][0]
    for temps in dico[annee]:
        if temps < meilleur:
            meilleur = temps
    return meilleur
```

👉*Commentaire : On parcourt tous les temps pour conserver le plus petit, correspondant au meilleur temps.*

---

12. Indiquer la valeur affectée à la variable `i_cat` au cours de l'appel `mystere(dict_perf_5km, 'SH').`

```
Indice 2
```

👉*Commentaire : Dans la liste des catégories `["JH", "JF", "SH", "SF", "MH", "MF"]`, la catégorie `SH` se trouve à l'indice 2.*

---

13. Indiquer le message d'erreur renvoyé pour `mystere(dict_perf_5km,'SG').`

```
"local variable 'i_cat' referenced before assignment"
```

👉*Commentaire : Comme `SG` n'existe pas dans les catégories, la variable `i_cat` n'est jamais définie avant son utilisation.*

---

14. Proposer une assertion à ajouter entre la ligne 2 et la ligne 3.

```
assert cat in ['JH', 'JF', 'SH', 'SF', 'MH', 'MF'], "Catégorie
invalide"
```

👉*Commentaire : L'assertion vérifie que la catégorie saisie appartient bien à la liste des catégories autorisées.*

---

15. Indiquer le résultat de l'appel `mystere(dict_perf_5km,'SH').`

```
[900, 1100, 1000, 1000]
```

👉*Commentaire : La fonction récupère les performances de la catégorie `SH` pour chaque année du dictionnaire.*

---

16. Écrire le code de la fonction `records`.

```
def records(dico):
    resultat = [86400, 86400, 86400, 86400, 86400, 86400]
    for annee in dico:
        for i in range(6):
            if dico[annee][i] < resultat[i]:
                resultat[i] = dico[annee][i]
    return resultat
```

👉*Commentaire : On initialise chaque record à une très grande valeur (24 h = 86400 s), puis on conserve le plus petit temps trouvé pour chaque catégorie.*
