

CORRECTION BACCALAURÉAT BLANC

Sujet 1

EXERCICE 1 (6 points)



Cet exercice porte sur l'architecture matérielle, les réseaux, les routeurs et les protocoles de routage

On considère un réseau local N1 constitué de trois ordinateurs M1, M2, M3 et dont les adresses IP sont les suivantes :

- M1 : 192.168.1.1/24;
- M2 : 192.168.1.2/24;
- M3 : 192.168.2.3/24.

On rappelle que le "/24" situé à la suite de l'adresse IP de M1 signifie que l'adresse réseau du réseau local N1 est 192.168.1.0.

Depuis l'ordinateur M1, un utilisateur exécute la commande ping vers l'ordinateur M3 comme suit :

```
1 util@M1 ~ % ping 192.168.2.3
2 PING 192.168.2.3 (192.168.2.3): 56 data bytes
3 Hôte inaccessible
```

1. Expliquer le résultat obtenu lors de l'utilisation de la commande ping (on part du principe que la connexion physique entre les machines est fonctionnelle).

Correction

L'adresse du réseau de l'ordinateur M1 est 192.168.1.0 et celle de l'ordinateur M3 est 192.168.2.0.

Comme les adresses réseau sont différentes, les deux ordinateurs ne communiquent pas entre-eux.

Il faudrait placer un routeur pour pouvoir les faire communiquer.

On ajoute un routeur R1 au réseau N1 :

"Un routeur moderne se présente comme un boîtier regroupant carte mère, microprocesseur, ROM, RAM ainsi que les ressources réseaux nécessaires (Wi-Fi, Ethernet...). On peut donc le voir comme un ordinateur minimal dédié, dont le système d'exploitation peut être un Linux allégé. De même, tout ordinateur disposant des interfaces adéquates (au minimum deux, souvent Ethernet) peut faire office de routeur s'il est correctement configuré (certaines distributions Linux minimales spécialisent la machine dans cette fonction)."

Source : Wikipédia, article "Routeur"

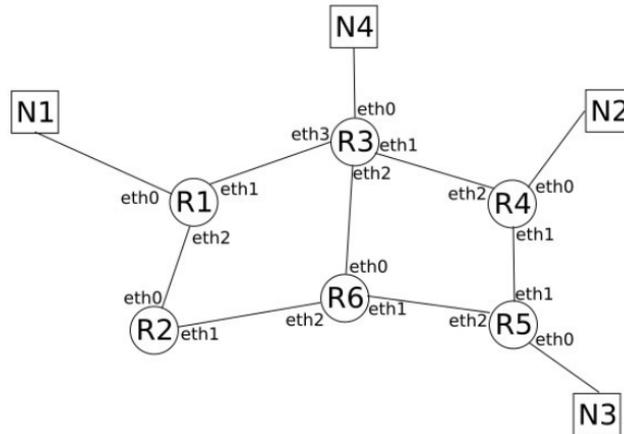
2. Expliquer ce qu'est **Linux**.

Correction

Linux est un système d'exploitation libre

Le réseau N1 est maintenant relié à d'autres réseaux locaux (N2, N3, N4) par l'intermédiaire d'une série de routeurs (R1, R2, R3, R4, R5, R6) :

Figure 1. Schéma du réseau



3. Attribuer une adresse IP valide à l'interface eth0 du routeur R1 sachant que l'adresse réseau du réseau N1 est 192.168.1.0.

Correction

L'adresse du réseau étant 192.168.1.0, on peut attribuer toutes les adresses **libre** sur la plage de 192.168.1.1 à 192.168.1.254, par exemple l'adresse 192.168.1.254.

Dans un premier temps, on utilise le protocole de routage **RIP** (Routing Information Protocol). On rappelle que dans ce protocole, la métrique de la table de routage correspond au nombre de routeurs à traverser pour atteindre la destination. La table de routage du routeur R1 est donnée dans le tableau suivant :

| Table de routage du routeur R1 | | |
|--------------------------------|---------------------|----------|
| destination | interface de sortie | métrique |
| N1 | eth0 | 0 |
| N2 | eth1 | 2 |
| N2 | eth2 | 4 |
| N3 | eth1 | 3 |
| N3 | eth2 | 3 |
| N4 | eth1 | 1 |
| N4 | eth2 | 3 |

4. Déterminer le chemin parcouru par un paquet de données pour aller d'une machine appartenant au réseau N1 à une machine appartenant au réseau N2.

Correction

D'après la table de routage, suivant le protocole RIP, le routeur **R1** va envoyer le paquet par l'interface **eth1** sur le routeur **R3** puis du **R3** ver le routeur **R4**.
On a donc : N1 - R1 - R3 - R4 - N2

Le routeur R3 tombe en panne. Après quelques minutes, la table de routage de R1 est modifiée afin de tenir compte de cette panne.

5. Dresser la table de routage du routeur R1 suite à la panne du routeur R3.

Correction

| Table de routage du routeur R1 | | |
|--------------------------------|---------------------|----------|
| destination | interface de sortie | métrique |
| N1 | eth0 | 0 |
| N2 | eth2 | 4 |
| N3 | eth2 | 3 |
| N4 | | |

Le routeur R3 est de nouveau fonctionnel. Dans la suite de cet exercice, on utilise le protocole de routage OSPF (Open Shortest Path First). On rappelle que dans ce protocole, la métrique de la table de routage correspond à la somme des coûts :

$$\text{coût} = \frac{10^8}{d} \quad (\text{où } d \text{ est la bande passante d'une liaison en bit/s})$$

. Le réseau est constitué de 3 types de liaison de communication :

- Fibre avec un débit de 1 Gbit/s;
- Fast-Ethernet avec un débit de 100 Mbit/s;
- Ethernet avec un débit de 10 Mbit/s.

6. Calculer le coût de chacune de ces liaisons de communication.

Correction

- Fibre : Coût = $\frac{10^8}{10^9} = 0,1$
- Fast-Ethernet : Coût = $\frac{10^8}{10^8} = 1$
- Ethernet : Coût = $\frac{10^8}{10^7} = 10$

La table de routage du routeur R1 est donnée dans le tableau suivant :

| Table de routage du routeur R1 | | |
|--------------------------------|---------------------|----------|
| destination | interface de sortie | métrique |
| N1 | eth0 | 0 |
| N2 | eth1 | 10,1 |
| N2 | eth2 | 1,3 |
| N3 | eth1 | 11,3 |
| N3 | eth2 | 0,3 |
| N4 | eth1 | 10 |
| N4 | eth2 | 1,2 |

D'autre part, le type des différentes liaisons inter-routeurs sont les suivantes :

- R1 - R2 : Fibre;
- R1 - R3 : Ethernet;
- R2 - R6 : INCONNU;
- R3 - R6 : Fast-Ethernet;
- R3 - R4 : Fibre;
- R4 - R5 : Fast-Ethernet;
- R5 - R6 : Fibre.

7. Dédurre de la table de routage de R1 et du schéma du réseau le type de la liaison inter-routeur R2 - R6.

Correction

Pour aller à R5 (N3) en passant par l'eth2 du routeur R1, le coût total est de 0,3, donc $\text{Cout}(R1 : R2) + \text{Cout}(R2 : R3) + \text{Cout}(R2 : R5) = 0,3$

$0,1 + \text{Cout}(R2 : R3) + 0,1 = 0,3$

$\text{Cout}(R2 : R3) = 0,1$

La liaison R2-R3 est donc en fibre.

Des travaux d'amélioration ont été réalisés sur ce réseau : la liaison inter-routeur R1-R3 est désormais de type Fibre.

8. Modifier la table de routage de R1 en tenant compte de cette amélioration.

Correction

| Table de routage du routeur R1 | | |
|--------------------------------|---------------------|------------|
| destination | interface de sortie | métrique |
| N1 | eth0 | 0 |
| N2 | eth1 | 0,2 |
| N2 | eth2 | 1,3 |
| N3 | eth1 | 1,4 |
| N3 | eth2 | 0,3 |
| N4 | eth1 | 0,1 |
| N4 | eth2 | 1,2 |

On ajoute un réseau local N5 et un routeur R7 au réseau étudié ci-dessus. Le routeur R7 possède trois interfaces réseaux eth0, eth1 et eth2. eth0 est directement relié au réseau local N5. eth1 et eth2 sont reliés à d'autres routeurs (ces liaisons inter-routeur sont de type Fibre).

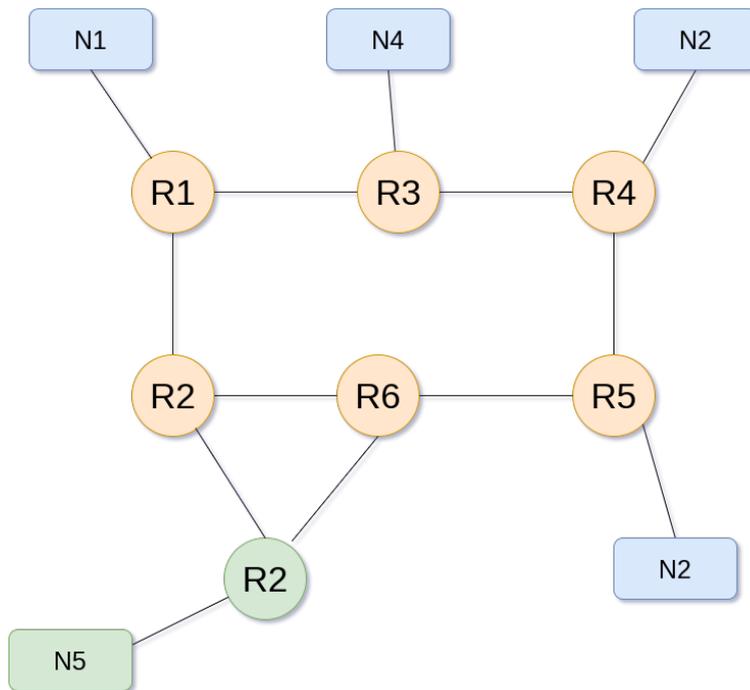
Les deux tableaux suivants présentent un extrait des tables de routage des routeurs R1 et R3 :

| Extrait Table de routage du routeur R1 | | |
|--|---------------------|----------|
| destination | interface de sortie | métrique |
| ... | ... | ... |
| N5 | eth1 | 1,2 |
| N5 | eth2 | 0,2 |

| Extrait Table de routage du routeur R1 | | |
|--|---------------------|----------|
| destination | interface de sortie | métrique |
| ... | ... | ... |
| N5 | eth1 | 1,3 |
| N5 | eth2 | 1,1 |
| N5 | eth3 | 0,3 |

9. Recopier et compléter le schéma du réseau (Figure. 1) en ajoutant le routeur R7 et le réseau local N5.

Correction



EXERCICE 2 (6 points)

i Cet exercice porte sur les arbres binaires, les files et la programmation orientée objet. Cet exercice comporte deux parties indépendantes.

Partie 1

Une entreprise stocke les identifiants de ses clients dans un arbre binaire de recherche. On rappelle qu'un arbre binaire est composé de nœuds, chacun des nœuds possédant éventuellement un sous-arbre gauche et éventuellement un sous-arbre droit.

La taille d'un arbre est le nombre de nœuds qu'il contient. Sa hauteur est le nombre de nœuds du plus long chemin qui joint le nœud racine à l'une des feuilles.

On convient que la hauteur d'un arbre ne contenant qu'un nœud vaut 1 et celle de l'arbre vide vaut 0.

Dans cet arbre binaire de recherche, chaque nœud contient une valeur, ici une chaîne de caractères, qui est, avec l'ordre lexicographique (celui du dictionnaire) :

- strictement supérieure à toutes les valeurs des nœuds du sous-arbre gauche ;
- strictement inférieure à toutes les valeurs des nœuds du sous-arbre droit.

Ainsi les valeurs de cet arbre sont toutes distinctes. On considère l'arbre binaire de recherche suivant :

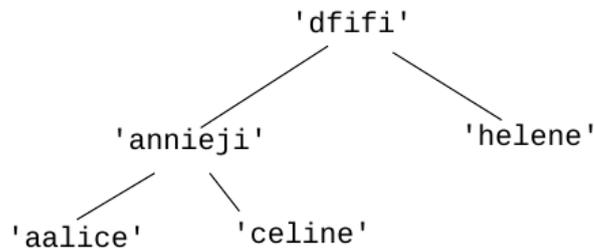


Figure 1. Arbre binaire de recherche.

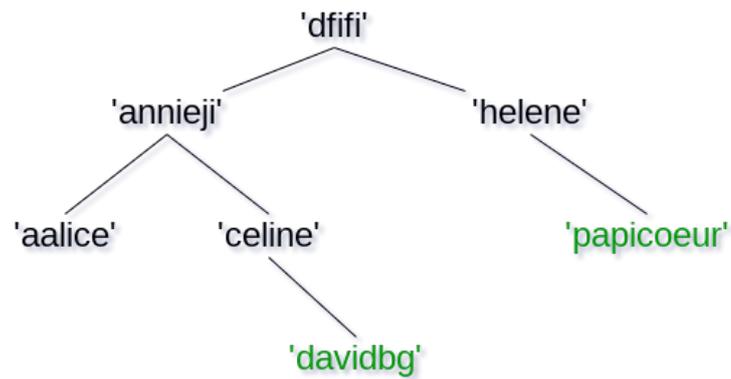
1. Donner la taille et la hauteur de l'arbre binaire de recherche de la figure 1.

Correction

- La taille de l'arbre binaire de recherche de la figure 1 est 5
- Sa hauteur est 3

2. Recopier cet arbre après l'ajout des identifiants suivants : 'davidbg' et 'papicoeur' dans cet ordre.

Correction



3. On décide de parcourir cet arbre pour obtenir la liste des identifiants dans l'ordre alphabétique.

Recopier la lettre correspondant au parcours à utiliser parmi les propositions suivantes :

- A - Parcours en largeur d'abord
- B - Parcours en profondeur dans l'ordre préfixe
- C - Parcours en profondeur dans l'ordre infixe
- D - Parcours en profondeur dans l'ordre suffixe (ou postfixe)

Correction

C - Parcours en profondeur dans l'ordre infixe

4. Pour traiter informatiquement les arbres binaires, nous allons utiliser une classe **ABR**.

Un arbre binaire de recherche, nommé **abr** dispose des méthodes suivantes :

- **abr.est_vide()** : renvoie **True** si **abr** est vide et **False** sinon.
- **abr.racine()** : renvoie l'élément situé à la racine de **abr** si **abr** n'est pas vide et **None** sinon.
- **abr.sg()** : renvoie le sous-arbre gauche de **abr** s'il existe et **None** sinon.
- **abr.sd()** : renvoie le sous-arbre droit de **abr** s'il existe et **None** sinon.

On a commencé à écrire une méthode récursive present de la classe **ABR**, où le paramètre identifiant est une chaîne de caractères et qui retourne **True** si identifiant est dans l'arbre et **False** sinon.

Correction

```

1     def present(self, identifiant):
2         if self.est_vide():
3             return False
4         elif self.racine() == identifiant:
5             return True
6         elif self.racine() < identifiant:
7             return self.sd().present(identifiant)
8         else:
9             return self.sg().present(identifiant)
10
  
```

Recopier et compléter les lignes 5, 7 et 9 de cette méthode.

Partie 2

On considère une structure de données file que l'on représentera par des éléments en ligne, l'élément à droite étant la tête de la file et l'élément à gauche étant la queue de la file.

On appellera **f1** la file suivante :

| | | | | | |
|--|-------|-------|--------|--------|--|
| | 'bac' | 'nsi' | '2023' | 'file' | |
|--|-------|-------|--------|--------|--|

On suppose que les quatre fonctions suivantes ont été programmées préalablement en langage Python :

- **creer_file()** : renvoie une file vide ;
- **est_vide(f)** : renvoie **True** si la file **f** est vide et **False** sinon ;
- **enfiler(f, e)** : ajoute l'élément **e** à la queue de la file **f** ;
- **defiler(f)** : renvoie l'élément situé à la tête de la file **f** et le retire de la file.

5. Les trois questions suivantes sont indépendantes.

- (a) Donner le résultat renvoyé après l'appel de la fonction **est_vide(f1)**.

Correction

La pile **f1** n'étant pas vide, le retour l'appel de la fonction sera **False**

- (b) Représenter la file **f1** après l'exécution du code **defiler(f1)**.

Correction

Après avoir utiliser la méthode **defiler**, la tête est enlever de la la file.
La représentation de **f1** est donc :

| | | | | |
|--|-------|-------|--------|--|
| | 'bac' | 'nsi' | '2023' | |
|--|-------|-------|--------|--|

- (c) Représenter la file **f2** après l'exécution du code suivant :

```
1 f2 = creer_file()
2 liste = ['castor', 'python', 'poule']
3 for elt in liste:
4     enfiler(f2, elt)
```

Correction

En respectant l'ordre dans lequel les éléments ont été enfilés, on obtient :

| | | | | |
|--|---------|----------|----------|--|
| | 'poule' | 'python' | 'castor' | |
|--|---------|----------|----------|--|

6. Recopier et compléter les lignes 4, 6 et 7 de la fonction `longueur` qui prend en paramètre une file `f` et qui renvoie le nombre d'éléments qu'elle contient. Après un appel à la fonction, la file `f` doit retrouver son état d'origine.

Correction

```
1 def longueur(f):
2     resultat = 0
3     g = creer_file()
4     while not f.est_vide() :
5         elt = defiler(f)
6         resultat = resultat +1
7         enfiler(g , elt)
8     while not(est_vide(g)):
9         enfiler(f, defiler(g))
10    return resultat
```

7. Un site impose à ses clients des critères sur leur mot de passe. Pour cela il utilise la fonction `est_valide` qui prend en paramètre une chaîne de caractères `mot` et qui retourne `True` si `mot` correspond aux critères et `False` sinon.

```
1 def est_valide(mot):
2     if len(mot) < 8:
3         return False
4     for c in mot:
5         if c in ['!', '#', '@', ';', ':']:
6             return True
7     return False
8
9
```

Parmi les mots de passe suivants, recopier celui qui sera validé par cette fonction.

- A - 'best@'
- B - 'paptap23'
- C - '2!@59fgds'

Correction

A est trop court

B ne contient de caractère spéciaux parmi '!', '@', ';' et ':'.

C'est donc C qui sera validé

EXERCICE 3 (8 points)

i Cet exercice porte sur la programmation Python (dictionnaire), la programmation orientée objet, les bases de données relationnelles et les requêtes SQL. Cet exercice est composé de 3 parties indépendantes.

On veut créer une application permettant de stocker et de traiter des informations sur des livres de science-fiction. On désire stocker les informations suivantes :

- l'identifiant du livre (id);
- le titre (titre);
- le nom de l'auteur (nom_auteur);
- l'année de première publication (ann_pub);
- une note sur 10 (note).

Voici un extrait des informations que l'on cherche à stocker :

| id | titre | auteur | ann_pub | note |
|----|-----------------------|----------|---------|------|
| 1 | 1984 | Orwell | 1949 | 10 |
| 2 | Dune | Herbert | 1965 | 9 |
| 14 | Fondation | Asimov | 1951 | 10 |
| 4 | Ubik | K.Dick | 1953 | 8 |
| 8 | Blade Runner | K.Dick | 1968 | 8 |
| 7 | Les Robots | Asimov | 1950 | 9 |
| 15 | Ravage | Barjavel | 1943 | 6 |
| 17 | Chroniques martiennes | Bradbury | 1950 | 7 |
| 9 | Dragon déchu | Hamilton | 2003 | 8 |
| 10 | Fahrenheit 451 | Bradbury | 1953 | 8 |

Partie A

Dans cette première partie, on utilise un dictionnaire Python. On considère le programme suivant :

```
1 dico_livres = {
2     'id' : [1, 2, 14, 4, 5, 8, 7, 15, 9, 10],
3     'titre' : ['1984', 'Dune', 'Fondation', 'Ubik', 'Blade Runner',
4               'Les Robots', 'Ravage', 'Chroniques martiennes',
5               'Dragon déchu', 'Fahrenheit 451'],
6     'auteur' : ['Orwell', 'Herbert', 'Asimov',
7                'K.Dick', 'K.Dick', 'Asimov', 'Barjavel',
```

```

8         'Bradbury', 'Hamilton', 'Bradbury'],
9     'ann_pub' : [1949,1965,1951,1953,1968,1950,1943,1950,2003,1953],
10    'note' : [10, 8, 9, 9, 8, 10, 6, 7, 8, 8]
11    }
12    a = dico_livres['note']
13    b = dico_livres['titre'][2]

```

1. Déterminer les valeurs des variables **a** et **b** après l'exécution de ce programme.

Correction

La valeur de **a** sera [10, 8, 9, 9, 8, 10, 6, 7, 8, 8]

La valeur de **b** sera **Fondation**

La fonction **titre_livre** prend en paramètre un dictionnaire (de même structure que **dico_livres**) et un identifiant, et renvoie le titre du livre qui correspond à cet identifiant.

Dans le cas où l'identifiant passé en paramètre n'est pas présent dans le dictionnaire, la fonction renvoie **None**.

2. Recopier et compléter les lignes 3, 4 et 5 de la fonction **titre_livre**.

Correction

```

1 def titre_livre(dico, id_livre):
2     for i in range(len(dico['id'])):
3         if dico['id'][i] == id_livre :
4             return dico['titre'][i]
5     return None

```

3. Écrire une fonction **note_maxi** qui prend en paramètre un dictionnaire **dico** (de même structure que **dico_livres**) et qui renvoie la note maximale.

Correction

```

1 def note_maxi(dico):
2     maxi = dico["note"][0]
3     for note in dico["note"]:
4         if note > maxi:
5             maxi = note
6     return maxi

```

4. Écrire une fonction **livres_note** qui prend en paramètre un dictionnaire **dico** (de même structure que **dico_livres**) et une note **n**, et qui renvoie la liste des titres des livres ayant obtenu la note **n** (on rappelle que **t.append(a)** permet de rajouter l'élément **a** à la fin de la liste **t**).

Correction

```

1 def livres\_note(dico,n):
2     livres = []
3     for i in range(len(dico[\"note\"])):
4         if dico[\"note\"][i] == n:
5             livres.append(dico[\"titre\"][i])
6     return livres

```

5. Écrire une fonction `livre_note_maxi` qui prend en paramètre un dictionnaire `dico` (de même structure que `dico_livres`) et qui renvoie la liste des titres des livres ayant obtenu la meilleure note sous la forme d'une liste Python.

Correction

```

1 def livre_note_maxi(dico):
2     note_max = note_maxi(dico)
3     livres = dico_livre(dico,note_max)
4     return livre

```

Partie B

Dans cette partie, on utilise le paradigme orientée objet (POO). On propose deux classes : `Livre` et `Bibliotheque`.

```

1 class Livre:
2     def __init__(self, id_livre, titre, auteur, ann_pub, note):
3         self.id = id_livre
4         self.titre = titre
5         self.auteur = auteur
6         self.ann_pub = ann_pub
7         self.note = note
8
9     def get_id(self):
10        return self.id
11
12    def get_titre(self):
13        return self.titre
14
15    def get_auteur(self):
16        return self.auteur
17
18    def get_ann_pub(self):
19        return self.ann_pub
20
21 class Bibliotheque:
22    def __init__(self):
23        self.liste_livre = []
24
25    def ajout_livre(self, livre):
26        self.liste_livre.append(livre)

```

```
27
28     def titre_livre(self, id_livre):
29         for livre in self.liste_livre :
30             if ... == id_livre :
31                 return ...
32     return ...
```

6. Citer un attribut et une méthode de la classe **Livre**.

_____ **Correction** _____

id est un attribut de la classe **Livre** et **get_titre** est une méthode de la classe **Livre**

7. Écrire la méthode **get_note** de la classe **Livre**. Cette méthode devra renvoyer la note d'un livre.

_____ **Correction** _____

```
1     def get_note(self):
2         return self.note
```

8. Écrire le programme permettant d'ajouter le livre **Blade Runner** à la fin de la **bibliothèque**, une instance de l'objet **Bibliothèque** déjà créée, en utilisant la classe **Livre** et la classe **Bibliothèque** (voir le tableau en début d'exercice).

_____ **Correction** _____

```
1     blade = Livre(8, "Blade Runner", "K.Dick", "1968", "8")
2     bibliotheque.ajouter_livre(blade)
```

9. Recopier et compléter la méthode **titre_livre** de la classe **Bibliothèque**. Cette méthode prend en paramètre l'identifiant d'un livre et renvoie le titre du livre si l'identifiant existe, ou **None** si l'identifiant n'existe pas.

_____ **Correction** _____

```
1     def titre_livre(self, id_livre):
2         for livre in self.liste_livre :
3             if livre.id == id_livre :
4                 return livre.titre
5     return None
```

Partie C

On utilise maintenant une base de données relationnelle. Les commandes nécessaires ont été exécutées afin de créer une table **livres**. Cette table **livres** contient toutes les données sur les livres. On obtient donc la table suivante :

| id | titre | auteur | ann_pub | note |
|----|-----------------------|----------|---------|------|
| 1 | 1984 | Orwell | 1949 | 10 |
| 2 | Dune | Herbert | 1965 | 9 |
| 14 | Fondation | Asimov | 1951 | 10 |
| 4 | Ubik | K.Dick | 1953 | 8 |
| 8 | Blade Runner | K.Dick | 1968 | 8 |
| 7 | Les Robots | Asimov | 1950 | 9 |
| 15 | Ravage | Barjavel | 1943 | 6 |
| 17 | Chroniques martiennes | Bradbury | 1950 | 7 |
| 9 | Dragon déchu | Hamilton | 2003 | 8 |
| 10 | Fahrenheit 451 | Bradbury | 1953 | 8 |

L'attribut id est la clé primaire pour la table livres.

10. Expliquer pourquoi l'attribut auteur ne peut pas être choisi comme clé primaire.

_____ **Correction** _____

L'attribut auteur ne peut pas être utilisé comme clé primaire car un même auteur peut écrire plusieurs livres.

11. Donner le résultat renvoyé par la requête SQL suivante :

```
1 SELECT titre
2 FROM livres
3 WHERE auteur = 'K.Dick';
```

_____ **Correction** _____

('Ubik'), ('Blade Runner')

12. Écrire une requête SQL permettant d'obtenir les titres des livres écrits par Asimov publiés après 1950.

_____ **Correction** _____

```
1 SELECT titre
2 FROM livres
3 WHERE auteur = 'Asimov' and ann_pub > 1950;
```

13. Écrire une requête SQL permettant de modifier la note du livre Ubik en la passant de 9/10 à 10/10.

Correction

```
1 UPDATE livres SET note = 10
2 WHERE id = 4
```

On souhaite proposer plus d'informations sur les auteurs des livres. Pour cela, on crée une deuxième table auteurs avec les attributs suivants :

- **id** de type INT;
- **nom** de type TEXT;
- **prenom** de type TEXT;
- **annee_naissance** de type INT (année de naissance).

auteurs

| id | nom | prenom | annee_naissance |
|----|----------|--------|-----------------|
| 1 | Orwell | George | 1903 |
| 2 | Herbert | Franck | 1920 |
| 3 | Asimov | Isaac | 1920 |
| 4 | K.Dick | Philip | 1928 |
| 5 | Bradbury | Ray | 1920 |
| 6 | Barjavel | René | 1911 |
| 7 | Hamilton | Peter | 1960 |

La table livres est aussi modifiée comme suit :

livres

| id | titre | id_auteur | ann_pub | note |
|----|-----------------------|-----------|---------|------|
| 1 | 1984 | 1 | 1949 | 10 |
| 2 | Dune | 2 | 1965 | 9 |
| 14 | Fondation | 3 | 1951 | 10 |
| 4 | Ubik | 4 | 1953 | 8 |
| 8 | Blade Runner | 4 | 1968 | 8 |
| 7 | Les Robots | 3 | 1950 | 9 |
| 15 | Ravage | 6 | 1943 | 6 |
| 17 | Chroniques martiennes | 5 | 1950 | 7 |
| 9 | Dragon déchu | 7 | 2003 | 8 |
| 10 | Fahrenheit 451 | 5 | 1953 | 8 |

14. Expliquer le rôle de l'attribut **id_auteur** de la table livres.

Correction

C'est une clé étrangère car il correspond à la clé primaire **id** de la table **auteurs**. Il permet donc de faire les jointures entre ces deux tables.

15. Écrire une requête SQL qui renvoie le nom et le prénom des auteurs des livres publiés après 1960.

Correction

```

1 SELECT auteurs.nom, auteurs.prenom FROM auteurs
2 JOIN livres ON livre.auteur_id = auteurs.id
3 WHERE livres.ann_pub > 1960;

```

16. Décrire par une phrase en français le résultat de la requête SQL suivante :

```

1 SELECT titre
2 FROM livres
3 JOIN auteurs ON livres.id_auteur = auteurs.id
4 WHERE livres.ann_pub - auteurs.annee_naissance < 30;

```

Correction

Cela renverra tous les titres des livres qui ont été publié par de auteurs qui avaient moins de 30 ans.