Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

CORRIGÉ NON OFFICIEL

Proposition de corrigé rédigée par FormaV avec l'aide d'outils d'intelligence artificielle. Elle peut contenir des erreurs et ne remplace ni le corrigé ni le barème officiels de l'examen.

Les sujets officiels des sessions passées et nos sujets d'entraînement sont classés sur https://bac-maths-nsi.fr/annales-sujets-et-corriges

Document mis en ligne par FormaV®. Reproduction ou diffusion sans autorisation interdite.

BACCALAURÉAT GÉNÉRAL • CORRIGÉ DÉTAILLÉ

Numérique et sciences informatiques

Session 2024 • Amérique du Nord • Jour 1

Code sujet **24-NSIJ1AN1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Programmation Python, files, ordonnancement des processus et interblocage
7 réponses • sujet p. 2–5

Exercice 2 — Graphes : matrice d’adjacence, dictionnaire de listes d’adjacence et parcours en profondeur
10 réponses • sujet p. 6–8

Exercice 3 — Flashcards : programmation Python, fichiers CSV, bases de données relationnelles et requêtes SQL
12 réponses • sujet p. 9–13

Les numéros, parties et pages renvoient au sujet original. Chaque réponse rappelle la consigne, explique la démarche et donne le résultat.

FormaV - document d'entraînement non officiel

Exercice 1

6 points

Programmation Python, files, ordonnancement des processus et interblocage

Question 1.

Sujet p. 2

> On s'intéresse à l'ordonnancement des processus dans un système d'exploitation utilisant un monoprocesseur. Citer les trois états dans lesquels un processus peut se trouver.

Dans un système d'exploitation, un processus peut attendre d'être exécuté, être en cours d'exécution ou être bloqué, par exemple en attente d'une ressource ou d'une entrée-sortie.

RÉPONSE À RETENIR

Les trois états sont : **prêt**, **en cours d'exécution** et **bloqué**.

Question 2.

Sujet p. 2

> Dans le contexte simplifié où l'on ne s'intéresse pas aux ressources qu'un processus pourrait acquérir ou libérer, citer les deux seuls états possibles pour un processus.

Dans le contexte simplifié de l'exercice, on ne modélise pas les attentes liées aux ressources. Un processus est donc soit prêt à recevoir le processeur, soit en cours d'exécution sur le processeur.

RÉPONSE À RETENIR

Les deux états possibles sont **prêt** et **en cours d'exécution**.

Question 3.

Sujet p. 2, 3

> Rectifier le code de la classe File pour que la méthode defile renvoie None lorsque la file est vide.

`pop(0)` provoque une erreur lorsque la liste est vide. Il faut donc tester la file avant de retirer son premier élément : si elle est vide, la méthode renvoie `None` ; sinon, elle renvoie l'élément retiré.

```
def defile(self):
    if self.est_vide():
        return None
    return self.contenu.pop(0)
```

RÉPONSE À RETENIR

Avec cette correction, `f.defile()` renvoie `None` si la file est vide. Si elle ne l'est pas, la méthode retire et renvoie le premier élément.

FormaV - document d'entraînement non officiel

Question 4.

Sujet p. 3, 4

> Recopier et compléter le chronogramme pour les processus p1, p2, p3 et p4, définis par p1 = Processus("p1", 4), p2 = Processus("p2", 3), p3 = Processus("p3", 5), p4 = Processus("p4", 3), avec depart_proc = {0: p1, 1: p3, 2: p2, 3: p4}.

À chaque cycle, on ajoute d'abord le processus créé à ce cycle, s'il y en a un. On exécute ensuite le premier processus de la file pendant un cycle. S'il n'est pas terminé, il est replacé en fin de file. Les intervalles ci-dessous sont de la forme [cycle, cycle + 1[; la file est indiquée de l'avant vers l'arrière, après l'exécution du cycle.

Cycle	Processus créé	Processus exécuté	File après le cycle
0	p1	p1	p1
1	p3	p1	p3, p1
2	p2	p3	p1, p2, p3
3	p4	p1	p2, p3, p4, p1
4	—	p2	p3, p4, p1, p2
5	—	p3	p4, p1, p2, p3
6	—	p4	p1, p2, p3, p4
7	—	p1	p2, p3, p4
8	—	p2	p3, p4, p2
9	—	p3	p4, p2, p3
10	—	p4	p2, p3, p4
11	—	p2	p3, p4
12	—	p3	p4, p3
13	—	p4	p3
14	—	p3	vide

Les processus reçoivent bien respectivement 4, 3, 5 et 3 cycles d'exécution, conformément à leurs durées.

> **RÉPONSE À RETENIR**
>
> Le chronogramme complété est : **0–1 : p1 ; 1–2 : p1 ; 2–3 : p3 ; 3–4 : p1 ; 4–5 : p2 ; 5–6 : p3 ; 6–7 : p4 ; 7–8 : p1 ; 8–9 : p2 ; 9–10 : p3 ; 10–11 : p4 ; 11–12 : p2 ; 12–13 : p3 ; 13–14 : p4 ; 14–15 : p3.**

Question 5.

Sujet p. 4

> Compléter le code incomplet de la classe Ordonnanceur pour mettre en place l'ordonnancement suivant la méthode du tourniquet.

La méthode `ajoute_nouveau_processus` ajoute le processus en fin de file. À chaque appel de `tourniquet`, on incrémente le temps, puis, si la file n'est pas vide, on retire le processus en tête et on l'exécute pendant un cycle. On le remet en fin de file seulement s'il n'est pas terminé.

```
class Ordonnanceur:

    def __init__(self):
        self.temps = 0
        self.file = File()

    def ajoute_nouveau_processus(self, proc):
        self.file.enfile(proc)

    def tourniquet(self):
        self.temps += 1
        if not self.file.est_vide():
            proc = self.file.defile()
            proc.execute_un_cycle()
            if not proc.est_fini():
                self.file.enfile(proc)
            return proc.nom
        else:
            return None
```

RÉPONSE À RETENIR

Le processus exécuté est retiré de la file, puis replacé en fin de file s'il reste des cycles à effectuer. Si la file est vide, `tourniquet` renvoie `None`.

Question 6.

Sujet p. 4, 5

> Écrire un programme qui utilise les variables p1, p2, p3, p4 et depart_proc définies précédemment, crée un ordonnanceur, ajoute un nouveau processus à l'ordonnanceur lorsque c'est le moment, affiche le processus choisi par l'ordonnanceur et s'arrête lorsqu'il n'y a plus de processus à exécuter.

On parcourt les cycles à partir de 0. À chaque cycle, on ajoute le processus associé dans `depart_proc`, s'il y en a un, avant d'appeler `tourniquet`. La boucle continue jusqu'à ce que toutes les créations aient eu lieu et que la file soit vide.

```
ordonnanceur = Ordonnanceur()
cycle = 0
dernier_depart = max(depart_proc)

while cycle <= dernier_depart or not ordonnanceur.file.est_vide():
    if cycle in depart_proc:
        ordonnanceur.ajoute_nouveau_processus(depart_proc[cycle])
    print(ordonnanceur.tourniquet())
    cycle += 1
```

Avec les données de l'énoncé, les processus sont créés aux cycles 0 (p1), 1 (p3), 2 (p2) et 3 (p4). La boucle s'arrête après le cycle où le dernier processus termine.

RÉPONSE À RETENIR

Le programme affiche successivement :

```
p1
p1
p3
p1
p2
p3
p4
p1
p2
p3
p4
p2
p3
p4
p3
```

Après le dernier affichage, la file est vide et toutes les créations prévues ont eu lieu : la boucle s’arrête.

Question 7.

Sujet p. 3, 5

> Dans la situation donnée en exemple (Figure 1), les processus A, B, C et D utilisent les ressources indiquées et effectuent les opérations précisées dans le tableau. Montrer que l’ordre d’exécution donné en exemple aboutit à une situation d’interblocage.

On suit l’ordre d’exécution de la Figure 1 et les opérations du tableau. Une ressource acquise est considérée comme détenue jusqu’à sa libération explicite.

- Au cycle 0, D acquiert le fichier et le conserve.
- Au cycle 2, B acquiert le clavier et le conserve.
- Au cycle 3, D demande le clavier, déjà détenu par B : D est alors bloqué tout en conservant le fichier.
- Au cycle 5, B demande le fichier, toujours détenu par D : B est alors bloqué tout en conservant le clavier.

Les deux processus sont donc en attente circulaire : D attend une ressource détenue par B, et B attend une ressource détenue par D. Aucun ne peut poursuivre pour libérer la ressource attendue par l’autre.

RÉPONSE À RETENIR

Il y a interblocage : **D détient le fichier et attend le clavier détenu par B ; B détient le clavier et attend le fichier détenu par D.**

Exercice 2

6 points

Graphes : matrice d’adjacence, dictionnaire de listes d’adjacence et parcours en profondeur

Partie A : Matrice d’adjacence

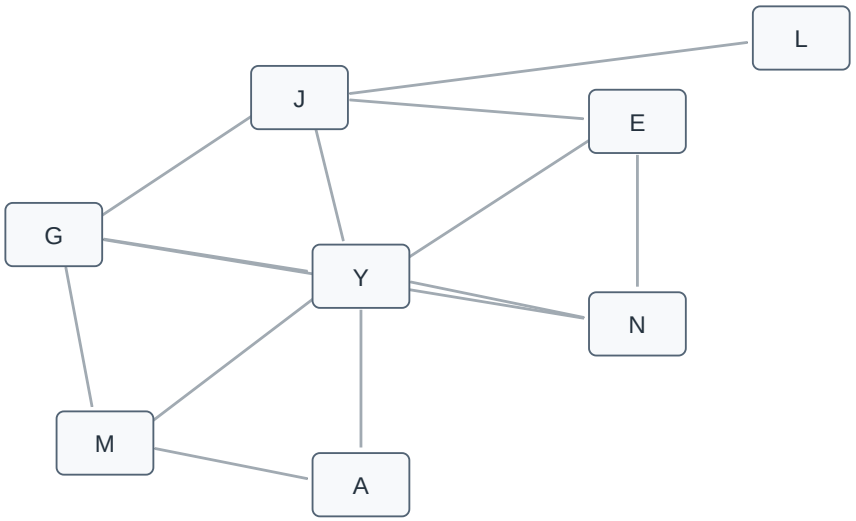
Question 1.

Sujet p. 6

> Dessiner le graphe des relations d’amitié en représentant chaque personne par la première lettre de son prénom entourée d’un cercle et chaque lien d’amitié par un trait entre deux personnes.

Le graphe est non orienté : une amitié entre deux personnes est représentée par un trait sans flèche. On ne représente chaque lien qu’une seule fois. Les huit sommets sont les initiales G, J, Y, E, N, M, A et L.

Graphe des relations d’amitié



G : Gabriel ; J : Jade ; Y : Yanis ; E : Emma ; N : Nina ; M : Milo ; A : Anas ; L : Lou.

RÉPONSE À RETENIR

Voici le graphe correspondant aux relations d’amitié de l’énoncé.

Question 2.

Sujet p. 6

> Recopier et compléter l’implémentation de la déclaration de la matrice d’adjacence du graphe, dans l’ordre des sommets G, J, Y, E, N, M, A, L.

Les lignes et les colonnes sont dans l’ordre G, J, Y, E, N, M, A, L. Une case vaut 1 si les deux personnes sont amies et 0 sinon. Le graphe étant non orienté, la matrice est symétrique. Par exemple, les amis de G sont J, Y, N et M : la première ligne contient donc des 1 aux colonnes correspondantes.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

La matrice complétée est :

```
matrice_adj = [[0, 1, 1, 0, 1, 1, 0, 0],  # G
               [1, 0, 1, 1, 0, 0, 0, 1],  # J
               [1, 1, 0, 1, 1, 1, 1, 0],  # Y
               [0, 1, 1, 0, 1, 0, 0, 0],  # E
               [1, 0, 1, 1, 0, 0, 0, 0],  # N
               [1, 0, 1, 0, 0, 0, 1, 0],  # M
               [0, 0, 1, 0, 0, 1, 0, 0],  # A
               [0, 1, 0, 0, 0, 0, 0, 0]]  # L
```

Question 3.

Sujet p. 7

> Indiquer les retours de l'exécution des instructions position(sommets, 'G') et position(sommets, 'Z'), avec sommets = ['G', 'J', 'Y', 'E', 'N', 'M', 'A', 'L'].

Les indices d'une liste Python commencent à 0. G est le premier élément de `sommets`, et Z n'y figure pas. La fonction `position` renvoie donc l'indice de G et `None` pour Z.

RÉPONSE À RETENIR

`position(sommets, 'G')` renvoie `0`.

`position(sommets, 'Z')` renvoie `None`.

Question 4.

Sujet p. 6, 7

> Recopier et compléter le code de la fonction nb_amis(L, m, s), qui prend en paramètres une liste de noms de sommets L, une matrice d'adjacence m et un nom de sommet s, et renvoie le nombre d'amis de s s'il est présent dans L, et None sinon.

On commence par chercher l'indice du sommet `s`. S'il n'est pas dans la liste, on renvoie `None`. Sinon, on additionne les valeurs de la ligne correspondante de la matrice : chaque 1 représente un ami et chaque 0 une absence de lien.

RÉPONSE À RETENIR

```
def nb_amis(L, m, s):
    pos_s = position(L, s)
    if pos_s is None:
        return None
    amis = 0
    for i in range(len(m)):
        amis += m[pos_s][i]
    return amis
```

Question 5.

Sujet p. 6, 7

> Indiquer le retour de l'exécution de la commande nb_amis(sommets, matrice_adj, 'G').

La ligne de G est `[0, 1, 1, 0, 1, 1, 0, 0]`. La fonction additionne ses valeurs : `0 + 1 + 1 + 0 + 1 + 1 + 0 + 0 = 4`.

> **RÉPONSE À RETENIR**
>
> `nb_amis(sommets, matrice_adj, 'G')` renvoie `4`.

Partie B : Dictionnaire de listes d'adjacence

Question 6.

Sujet p. 7

> Dans un dictionnaire Python {c : v}, indiquer ce que représentent c et v.

Dans une association de dictionnaire `{c : v}`, `c` est la clé et `v` est la valeur associée. Ici, la clé désigne un sommet et la valeur est la liste des sommets qui lui sont reliés par une arête.

> **RÉPONSE À RETENIR**
>
> `c` représente le nom du sommet ; `v` représente la liste de ses amis.

Question 7.

Sujet p. 6, 7

> Recopier et compléter le dictionnaire de listes d'adjacence graphe pour qu'il modélise complètement le groupe d'amis.

Chaque clé correspond à une personne et sa liste contient ses amis. Les listes sont réciproques : si J est dans la liste de G, alors G figure dans celle de J.

> **RÉPONSE À RETENIR**

```
graphe = {'G': ['J', 'Y', 'N', 'M'],
          'J': ['G', 'Y', 'E', 'L'],
          'Y': ['G', 'J', 'E', 'N', 'M', 'A'],
          'E': ['J', 'Y', 'N'],
          'N': ['G', 'Y', 'E'],
          'M': ['G', 'Y', 'A'],
          'A': ['Y', 'M'],
          'L': ['J']}
```

Question 8.

Sujet p. 7

> Écrire le code de la fonction nb_amis(d, s), qui prend en paramètres un dictionnaire d'adjacence d et un nom de sommet s, et renvoie le nombre d'amis de s. On suppose que s est bien dans d.

Les amis de `s` sont les éléments de la liste associée à la clé `s`. Leur nombre est donc `len(d[s])`. L'énoncé garantit que `s` est présent dans le dictionnaire.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
def nb_amis(d, s):
    return len(d[s])
```

Exemple : la liste de L est `['J']`, donc `nb_amis(graphe, 'L')` renvoie `1`.

Question 9.

Sujet p. 8

Après que Milo s'est fâché avec Gabriel et Yanis et qu'Anas s'est fâché avec Yanis, le dictionnaire d'adjacence du graphe est celui donné. Donner la liste du cercle d'amis de Lou, c'est-à-dire de toute personne atteignable dans le graphe à partir de Lou.

On suit le dictionnaire modifié fourni en page 8, en partant de L et en explorant les voisins dans l'ordre indiqué. On atteint J, puis G, N, Y et E. Les sommets M et A forment un autre groupe et ne sont pas atteignables depuis L. À noter : le dictionnaire fourni omet le lien G-Y, bien que ce lien figure dans les relations initiales et ne soit pas supprimé par les disputes décrites. On utilise ici le dictionnaire explicite de la question ; cette omission ne change ni l'ensemble atteignable depuis L ni l'ordre de parcours indiqué.

RÉPONSE À RETENIR

Le cercle d'amis de Lou, en incluant Lou comme sommet de départ, est constitué de L, J, G, N, Y et E, soit Lou, Jade, Gabriel, Nina, Yanis et Emma.

Ordre	Sommet atteint	Commentaire
1	L (Lou)	Sommet de départ
2	J (Jade)	Voisine de L
3	G (Gabriel)	Voisin non visité de J
4	N (Nina)	Voisin non visité de G
5	Y (Yanis)	Voisin non visité de N
6	E (Emma)	Voisine non visitée de Y

Question 10.

Sujet p. 8

Recopier et compléter le code de la fonction parcours_en_profondeur(d, s), qui prend en paramètres un dictionnaire d'adjacence d et un sommet s, et renvoie la liste des sommets issue du parcours en profondeur du graphe modélisé par d à partir de s.

On ajoute le sommet de départ à la liste des sommets visités, puis on explore récursivement ses voisins qui n'ont pas encore été visités. Transmettre la même liste `visites` à chaque appel permet de mémoriser les sommets déjà parcourus et d'éviter les boucles dues aux cycles. Le paramètre vaut `None` par défaut afin de ne pas partager accidentellement une même liste entre plusieurs appels indépendants.

Trace avec le graphe modifié de la question 9, au départ de L :

FormaV - document d'entraînement non officiel

Appel récursif	Liste après ajout du sommet
L	[L]
J	[L, J]
G	[L, J, G]
N	[L, J, G, N]
Y	[L, J, G, N, Y]
E	[L, J, G, N, Y, E]

Le parcours termine car chaque sommet nouvellement rencontré est ajouté à `visites` et ne peut être ajouté une seconde fois. Avec une liste pour tester l'appartenance, le coût est au plus de l'ordre de `V(V + E)`, où V est le nombre de sommets et E le nombre d'arêtes.

RÉPONSE À RETENIR

```
def parcours_en_profondeur(d, s, visites=None):
    if visites is None:
        visites = []
    if s not in visites:
        visites.append(s)
    for v in d[s]:
        if v not in visites:
            parcours_en_profondeur(d, v, visites)
    return visites
```

Sur le dictionnaire modifié, `parcours_en_profondeur(graphe, 'L')` renvoie `['L', 'J', 'G', 'N', 'Y', 'E']`.

Exercice 3

8 points

Flashcards : programmation Python, fichiers CSV, bases de données relationnelles et requêtes SQL

Partie A

Question 1.

Sujet p. 9, 10

> Donner le séparateur choisi par l'étudiante pour son fichier flashcards.csv.

Dans l'extrait, les champs d'une ligne sont séparés par des points-virgules : par exemple, l'en-tête est `discipline;chapitre;question;réponse`.

RÉPONSE À RETENIR

Le séparateur choisi est le point-virgule (`;`).

Question 2.

Sujet p. 9, 10

> Justifier pourquoi l'étudiante a choisi ce séparateur.

Le point-virgule sépare les champs sans couper les réponses qui contiennent des virgules. Par exemple, *Allemagne, Italie, Japon* doit rester une seule valeur dans le champ `réponse`.

RÉPONSE À RETENIR

L'étudiante a choisi le point-virgule pour que les virgules présentes dans certaines réponses ne soient pas prises pour des séparateurs de champs.

Question 3.

Sujet p. 9, 10, 11

> Recopier et compléter le code de la fonction charger(nom_fichier), qui lit le fichier dont le nom est fourni en argument et renvoie les données lues sous la forme d'un dictionnaire comme dans l'exemple fourni précédemment.

`csv.DictReader` utilise la première ligne comme noms de champs et crée un dictionnaire pour chaque ligne du fichier. Il faut indiquer le séparateur réellement présent dans le fichier, soit `;`, puis convertir les données en liste avant de les renvoyer.

```
def charger(nom_fichier):
    with open(nom_fichier, 'r', encoding='utf-8', newline='') as fichier:
        donnees = list(csv.DictReader(fichier, delimiter=';'))
    return donnees
```

Les valeurs lues par `DictReader` sont des chaînes de caractères.

RÉPONSE À RETENIR

```
def charger(nom_fichier):
    with open(nom_fichier, 'r', encoding='utf-8', newline='') as fichier:
        donnees = list(csv.DictReader(fichier, delimiter=';'))
    return donnees
```


FormaV - document d'entraînement non officiel

Question 4.

Sujet p. 10, 11

> Le module time est importé à la ligne 2 de ce programme. Quelle est la méthode du module time utilisée dans ce code ?

Le code appelle `time.sleep(5)` puis `time.sleep(1)`. La méthode `sleep` du module `time` suspend l'exécution pendant le nombre de secondes indiqué.

> **RÉPONSE À RETENIR**
>
> La méthode utilisée est `time.sleep`.

Question 5.

Sujet p. 10, 11

> Donner le type de la variable donnees[i] (par exemple ligne 12).

Chaque élément de la liste `donnees` correspond à une ligne lue par `csv.DictReader`. Cette ligne est représentée par un dictionnaire dont les clés sont les noms des champs de l'en-tête.

> **RÉPONSE À RETENIR**
>
> Le type de `donnees[i]` est `dict` (un dictionnaire Python).

Question 6.

Sujet p. 10, 11

> Recopier et compléter les lignes 41 à 44 du programme.

On charge le fichier dans `flashcard`, puis on transmet cette liste aux fonctions de sélection et d'entraînement. La fonction `choix_chapitre` du sujet fait référence à la variable globale `flashcard` dans son corps ; l'initialiser avant de l'appeler permet donc au code fourni de fonctionner.

```
flashcard = charger('flashcards.csv')
d = choix_discipline(flashcard)
c = choix_chapitre(flashcard, d)
entrainement(flashcard, d, c)
```

Pour rendre `choix_chapitre` plus modulaire, on pourrait aussi corriger son corps en remplaçant ses références à `flashcard[i]` par `donnees[i]`.

> **RÉPONSE À RETENIR**
>
> ```
> flashcard = charger('flashcards.csv')
> d = choix_discipline(flashcard)
> c = choix_chapitre(flashcard, d)
> entrainement(flashcard, d, c)
> ```

Partie B

Question 7.

Sujet p. 12

> Écrire une requête SQL qui complète la table boite et insère la boite 5 de libellé « tous les quinze jours » et de

FormaV - document d'entraînement non officiel

> fréquence 15.

La table `boite` possède les attributs `id`, `lib` et `frequence`. On insère donc l'identifiant 5, le libellé demandé et la fréquence 15 jours.

```
INSERT INTO boite (id, lib, frequence)
VALUES (5, 'tous les quinze jours', 15);
```

RÉPONSE À RETENIR

```
INSERT INTO boite (id, lib, frequence)
VALUES (5, 'tous les quinze jours', 15);
```

Question 8.

Sujet p. 12, 13

> Une requête sur la table flashcard affiche l'enregistrement 5, 2, 1, Pearl Harbor – date, 6 décembre 1941. Écrire une requête SQL pour mettre à jour la date de Pearl Harbor renvoyée. La bonne date est le 7 décembre 1941.

La valeur à corriger est la réponse associée à la flashcard d'identifiant 5 : le texte affiché est *6 décembre 1941*, alors que la réponse correcte est *7 décembre 1941*. L'attribut `date_interro` désigne, lui, la date de la prochaine interrogation ; ce n'est pas la date de l'événement à corriger. On sélectionne la ligne indiquée par sa clé primaire `id`.

```
UPDATE flashcard
SET reponse = '7 décembre 1941'
WHERE id = 5;
```

RÉPONSE À RETENIR

```
UPDATE flashcard
SET reponse = '7 décembre 1941'
WHERE id = 5;
```

Question 9.

Sujet p. 11, 13

> Écrire une requête SQL qui permet d'obtenir la liste des libellés des disciplines.

Les libellés des disciplines sont stockés dans l'attribut `lib` de la table `discipline`. La requête sélectionne cet attribut pour toutes les lignes de la table.

```
SELECT lib
FROM discipline;
```

Les deux libellés de discipline visibles dans l'extrait CSV sont `histoire` et `philosophie`. L'extrait ne donne pas les enregistrements de la table relationnelle : il ne permet donc pas d'affirmer que le résultat de la requête sur la base se limite à ces deux valeurs.

Libellé visible dans l'extrait CSV
histoire
philosophie

RÉPONSE À RETENIR

```
SELECT lib
FROM discipline;
```

Question 10.

Sujet p. 11, 12, 13

> Écrire une requête SQL qui permet d'obtenir la liste des libellés des chapitres de la discipline « histoire ».

Chaque chapitre indique la discipline à laquelle il appartient avec `id_disc`, qui correspond à `discipline.id`. On joint donc les deux tables, puis on ne conserve que les lignes dont le libellé de discipline est `histoire`.

```
SELECT chapitre.lib
FROM chapitre
JOIN discipline ON chapitre.id_disc = discipline.id
WHERE discipline.lib = 'histoire';
```

Les libellés de chapitres associés à *histoire* dans l'extrait CSV sont `crise de 1929` et `2GM`. Cet extrait n'est pas un relevé des enregistrements des tables relationnelles ; il ne suffit pas à établir le résultat complet de la requête sur la base.

Chapitre visible dans l'extrait CSV pour histoire
crise de 1929
2GM

RÉPONSE À RETENIR

```
SELECT chapitre.lib
FROM chapitre
JOIN discipline ON chapitre.id_disc = discipline.id
WHERE discipline.lib = 'histoire';
```

Question 11.

Sujet p. 11, 12, 13

> Écrire une requête SQL qui permet d'obtenir la liste des identifiants des flashcards de la discipline « histoire ».

Pour retrouver les flashcards de la discipline *histoire*, on relie chaque flashcard à son chapitre avec `flashcard.id_ch = chapitre.id`, puis ce chapitre à sa discipline avec `chapitre.id_disc = discipline.id`. On sélectionne ensuite les identifiants des flashcards et on filtre sur le libellé de la discipline.

```
SELECT flashcard.id
FROM flashcard
JOIN chapitre ON flashcard.id_ch = chapitre.id
JOIN discipline ON chapitre.id_disc = discipline.id
WHERE discipline.lib = 'histoire';
```

L'extrait CSV indique les questions et les réponses, mais ne fournit ni les identifiants des flashcards ni les enregistrements des tables relationnelles nécessaires aux jointures. Les identifiants renvoyés ne peuvent donc pas être déterminés à partir des données fournies.

RÉPONSE À RETENIR

```
SELECT flashcard.id
FROM flashcard
JOIN chapitre ON flashcard.id_ch = chapitre.id
JOIN discipline ON chapitre.id_disc = discipline.id
WHERE discipline.lib = 'histoire';
```

Le résultat des identifiants n'est pas déterminable avec les données fournies.

Question 12.

Sujet p. 12, 13

> Écrire une requête SQL pour supprimer toutes les flashcards de la boite d'identifiant 3.

On supprime les lignes de `flashcard` dont l'attribut `id_boite` vaut 3. La condition s'applique à toutes les flashcards de cette boîte.

```
DELETE FROM flashcard
WHERE id_boite = 3;
```

RÉPONSE À RETENIR

```
DELETE FROM flashcard
WHERE id_boite = 3;
```

Sujet : https://www.sujetdebac.fr/annales-pdf/2024/spe-numerique-informatique-2024-amerique-nord-1-sujet-officiel.pdf
© FormaV EI • Proposition de correction, distincte d'un corrigé officiel.

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.