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 • Métropole • Jour 2

Code sujet **24-NSIJ2ME1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Bases de données relationnelles, SQL et protocoles de sécurisation
9 réponses • sujet p. 2–4

Exercice 2 — Programmation orientée objets, tris, algorithmes gloutons et récursivité
10 réponses • sujet p. 5–8

Exercice 3 — Programmation orientée objet, classes Piste et Domaine, et recherche de chemins dans un graphe
16 réponses • sujet p. 9–15

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

Bases de données relationnelles, SQL et protocoles de sécurisation

Partie A – Bases de données

Question 1.

Sujet p. 2

> Indiquer, avec justification, s'il aurait été possible de choisir l'attribut Nom_artiste comme clé primaire dans la relation CD.

Une clé primaire doit identifier chaque ligne de façon unique. Or un artiste peut avoir plusieurs albums : son nom peut donc apparaître dans plusieurs lignes de `CD`. Le fait que chaque nom d'artiste n'apparaisse qu'une fois dans l'extrait fourni ne garantit pas que ce sera le cas dans toute la base.

RÉPONSE À RETENIR

Non. `Nom_artiste` ne convient pas comme clé primaire de `CD`, car un artiste peut posséder plusieurs albums. La clé primaire de `CD` est `id_album`.

Question 2.

Sujet p. 3, 4

> Écrire ce que renvoie la requête SELECT Nom_artiste FROM Artiste WHERE Pays = "Finlande" lorsqu'on l'applique aux extraits des relations.

La condition sélectionne les lignes de `Artiste` dont le pays est `Finlande`, puis la requête ne conserve que leur nom. Dans l'extrait, ces artistes sont Nightwish et The Rasmus. En l'absence de `ORDER BY`, l'ordre d'affichage n'est pas garanti.

RÉPONSE À RETENIR

Nom_artiste
Nightwish
The Rasmus

Question 3.

Sujet p. 3, 4

> Écrire ce que renvoie la requête SELECT CD.Annee FROM CD JOIN Artiste ON CD.Nom_artiste = Artiste.Nom_artiste WHERE Artiste.Style = "Metal".

La jointure associe chaque album à l'artiste ayant le même `Nom_artiste`. Le filtre conserve les artistes de style `Metal` : Metallica, Nightwish et Iron Maiden. Leurs albums de l'extrait sont sortis respectivement en 1986, 2001 et 1986. La requête ne comporte pas de `ORDER BY` : l'ordre des lignes n'est donc pas garanti.

RÉPONSE À RETENIR

Annee
1986
2001
1986

Question 4.

Sujet p. 3, 4

> Bob se rend compte que l'album Wishmaster est en réalité sorti en 2000. Donner la requête qu'il doit écrire pour mettre à jour sa base de données.

Il faut modifier l'attribut `Annee` de la ligne dont le titre d'album est *Wishmaster*. La condition sur le titre permet de cibler cette ligne.

RÉPONSE À RETENIR

```
UPDATE CD
SET Annee = 2000
WHERE Titre_album = 'Wishmaster';
```

Question 5.

Sujet p. 2, 4

> Donner la requête que Bob doit écrire pour afficher les titres de tous les albums de « Metal » rangés sur l'étagère dont le numéro est 1.

On joint `CD` à `Artiste` pour connaître le style, puis à `Rangement` pour connaître le numéro d'étagère. On sélectionne enfin les titres répondant aux deux conditions.

RÉPONSE À RETENIR

```
SELECT CD.Titre_album
FROM CD
JOIN Artiste
ON CD.Nom_artiste = Artiste.Nom_artiste
JOIN Rangement
ON CD.id_album = Rangement.id_album
WHERE Artiste.Style = 'Metal'
AND Rangement.Numero_etagere = 1;
```

Dans l'extrait, l'étagère 1 contient les albums d'Eminem et de Foo Fighters, de styles Rap et Rock. Aucun album de style Metal ne s'y trouve : le résultat est vide.

Titre_album

Question 6.

Sujet p. 2, 3, 4

> Bob a vendu l'album Dead Letters du groupe The Rasmus. Comme c'était le seul album de ce groupe qu'il possédait, il veut supprimer tous les enregistrements qui sont à présent inutiles dans les trois relations. Donner l'ordre dans lequel il doit les supprimer en expliquant pourquoi, puis écrire la requête correspondant à la suppression de l'album dans la relation CD.

Il faut supprimer les enregistrements dans l'ordre `Rangement`, puis `CD`, puis `Artiste`. En effet, `Rangement.id_album` référence `CD.id_album` : il faut donc retirer le rangement avant l'album. Ensuite, `CD.Nom_artiste` référence `Artiste.Nom_artiste` : il faut supprimer l'album avant l'artiste. Dans l'extrait, *Dead Letters* a l'identifiant 5 et *The Rasmus* n'a pas d'autre album.

> **RÉPONSE À RETENIR**
>
> Les suppressions sont donc :
>
> ```
> DELETE FROM Rangement
> WHERE id_album = 5;
>
> DELETE FROM CD
> WHERE id_album = 5;
>
> DELETE FROM Artiste
> WHERE Nom_artiste = 'The Rasmus';
> ```
>
> La requête demandée pour supprimer l'album de la relation `CD` est la deuxième.

Partie B – Sécurisation

Question 7.

Sujet p. 4

> Expliquer brièvement ce qu'est un algorithme de chiffrement symétrique.

Dans un chiffrement symétrique, les deux correspondants utilisent une même clé secrète : elle sert au chiffrement et au déchiffrement. Ils doivent donc avoir obtenu cette clé de manière sûre.

> **RÉPONSE À RETENIR**
>
> Un algorithme de chiffrement symétrique utilise la même clé secrète pour chiffrer et déchiffrer.

Question 8.

Sujet p. 4

> Rappeler brièvement ce qu'est un algorithme de chiffrement asymétrique.

Un chiffrement asymétrique repose sur une paire de clés liées : une clé publique, qui peut être diffusée, et une clé privée, qui doit rester secrète. Pour envoyer confidentiellement un message au détenteur de la paire, l'expéditeur le chiffre avec la clé publique de ce destinataire, qui le déchiffre avec sa clé privée.

> **RÉPONSE À RETENIR**
>
> Un algorithme de chiffrement asymétrique utilise une clé publique et une clé privée associées : le chiffrement destiné au détenteur se fait avec sa clé publique et le déchiffrement avec sa clé privée.

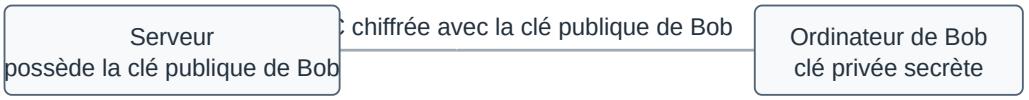
Question 9.

Sujet p. 4

> Bob possède une clé publique et une clé privée, et la clé publique de Bob est supposée connue par le serveur. Proposer une solution pour que le serveur puisse envoyer la clé C à l'ordinateur de Bob de façon sécurisée, c'est-à-dire pour que seul Bob puisse déchiffrer la clé envoyée.

Le serveur chiffre c avec la clé publique de Bob et envoie le résultat à son ordinateur. Bob déchiffre le message reçu avec sa clé privée. La confidentialité repose sur l'hypothèse que cette clé privée reste secrète et que la clé publique utilisée est bien celle de Bob. Ce chiffrement seul ne garantit ni l'authenticité de l'expéditeur ni l'intégrité du message.

Envoi confidentiel de la clé C à Bob



Bob déchiffre le message reçu avec sa clé privée et retrouve C.

RÉPONSE À RETENIR

Le serveur envoie à Bob c chiffrée avec la clé publique de Bob. Bob la déchiffre avec sa clé privée et retrouve c. Sous l'hypothèse que seul Bob détient cette clé privée et que la clé publique connue du serveur est bien celle de Bob, seul Bob peut déchiffrer la clé envoyée.

FormaV - document d'entraînement non officiel

Exercice 2

6 points

Programmation orientée objets, tris, algorithmes gloutons et récursivité

Partie A – Quelques outils

Question 1.

Sujet p. 5

> Compléter le constructeur qui renvoie un objet Marchandise. Utiliser le mot-clé assert afin qu'une exception soit levée si le paramètre v n'est pas strictement positif. Le constructeur donné est __init__(self, p: int, v: int) -> 'Marchandise'.

Le constructeur vérifie d'abord que le volume est strictement positif, puis affecte à l'objet ses deux attributs `prix` et `volume`.

```
class Marchandise:
    def __init__(self, p: int, v: int) -> 'Marchandise':
        assert v > 0
        self.prix = p
        self.volume = v
```

Si `v` est nul ou négatif, la condition de l'assertion est fausse et une exception est déclenchée.

> **RÉPONSE À RETENIR**
>
> Le constructeur complet est donné ci-dessus : il vérifie `v > 0` et initialise les attributs `prix` et `volume`.

Question 2.

Sujet p. 5

> Donner une instruction qui permet de créer une variable m1 représentant une marchandise d'un volume de 7 litres coûtant 20 €.

Le constructeur prend le prix en premier et le volume en second. On passe donc les valeurs 20 et 7 dans cet ordre.

```
m1 = Marchandise(20, 7)
```

> **RÉPONSE À RETENIR**
>
> `m1 = Marchandise(20, 7)`

Question 3.

Sujet p. 5

> Proposer une méthode ratio(self) -> float qui renvoie le ratio prix/volume d'une marchandise.

Le ratio demandé est le prix divisé par le volume. Le volume étant strictement positif, cette division est définie.

```
def ratio(self) -> float:
    return self.prix / self.volume
```


FormaV - document d'entraînement non officiel

Par exemple, pour `m1 = Marchandise(20, 7)`, la méthode renvoie le quotient exact 20/7, soit environ 2,857 euros par litre.

> **RÉPONSE À RETENIR**
>
> La méthode renvoie `self.prix / self.volume`.

Question 4.

Sujet p. 5

> Proposer une fonction prixListe(tab: list) -> int qui renvoie le prix cumulé de l'ensemble des marchandises formant le tableau tab.

On initialise un accumulateur à zéro, puis on ajoute le prix de chaque marchandise du tableau. Si le tableau est vide, la fonction renvoie donc zéro.

```
def prixListe(tab: list) -> int:
    total = 0
    for marchandise in tab:
        total = total + marchandise.prix
    return total
```

> **RÉPONSE À RETENIR**
>
> `prixListe` renvoie la somme des attributs `prix` des marchandises de `tab`.

Partie B – Première approche de rangement

Question 5.

Sujet p. 6

> Pour les quatre marchandises m_1 = (40, 20), m_2 = (210, 70), m_3 = (160, 40) et m_4 = (50, 50), préciser toutes les combinaisons de marchandises possibles si l'on ne dépasse pas un volume de 100 litres et le prix associé. En déduire la combinaison de marchandises qui maximise le prix.

Les marchandises étant indivisibles, chaque combinaison correspond à un sous-ensemble des quatre marchandises. On additionne les volumes et les prix et on conserve les combinaisons dont le volume total ne dépasse pas 100 litres. Le tableau inclut le chargement vide.

Combinaison	Volume (L)	Prix (€)
Aucune marchandise	0	0
m_1	20	40
m_2	70	210
m_3	40	160
m_4	50	50
m_1 et m_2	90	250
m_1 et m_3	60	200
m_1 et m_4	70	90
m_3 et m_4	90	210

Les deux autres paires dépassent la capacité : m_2 et m_3 occupent 110 litres, tandis que m_2 et m_4 en occupent 120. Toute combinaison de trois marchandises ou plus dépasse aussi 100 litres : la moins volumineuse des combinaisons de trois, m_1, m_3 et m_4, occupe déjà 110 litres.

RÉPONSE À RETENIR

Le prix maximal est **250 €**, obtenu avec la combinaison **m_1 et m_2**, qui occupe 20 + 70 = 90 litres.

Question 6.

Sujet p. 6

Parmi les qualificatifs suivants — dichotomique, glouton, graphique, insertion, maximum, récursif, tri — indiquer, sans justification, celui qui s'applique le mieux à l'algorithme précédent, ChargementGlouton.

L'algorithme choisit en priorité les marchandises ayant le plus grand ratio prix/volume.

RÉPONSE À RETENIR

Glouton.

Question 7.

Sujet p. 7

Recopier et compléter la fonction tri(tab: list) -> None afin qu'elle trie en place un tableau contenant des objets de type Marchandise selon l'ordre décroissant des ratios. Après l'appel tri(tab), tab[0] doit contenir la marchandise avec le plus haut ratio prix/volume.

Il s'agit d'un tri par insertion. À chaque étape, on mémorise la marchandise à insérer, puis on décale vers la droite les marchandises précédentes dont le ratio est inférieur. La condition `j >= 0` évite de dépasser le début du tableau.

FormaV - document d'entraînement non officiel

```
def tri(tab: list) -> None:
    n = len(tab)
    for i in range(1, n):
        marchandise = tab[i]
        j = i - 1
        while j >= 0 and tab[j].ratio() < marchandise.ratio():
            tab[j + 1] = tab[j]
            j = j - 1
        tab[j + 1] = marchandise
```

Trace avec les marchandises de la partie B : les ratios sont respectivement 2, 3, 4 et 1 pour m_1, m_2, m_3 et m_4. En partant de [m_1, m_2, m_3, m_4], l'insertion de m_2 donne [m_2, m_1, m_3, m_4]. L'insertion de m_3 donne [m_3, m_2, m_1, m_4]. Enfin, m_4 reste à sa place. Le tableau final est donc [m_3, m_2, m_1, m_4].

> **RÉPONSE À RETENIR**
>
> La fonction trie le tableau en place par ordre décroissant des ratios et ne renvoie pas de valeur.

Question 8.

Sujet p. 7

> Sans justifier, préciser le nom de ce tri, ainsi que son coût temporel dans le pire des cas, à choisir parmi constant, logarithmique, linéaire, quasi-linéaire ($n \log_2 n$), quadratique, cubique ou exponentiel.

Le code réalise un tri par insertion. Dans le pire des cas, chaque élément peut être comparé aux éléments précédents et décalé à travers tout le début du tableau. Le nombre d'opérations est alors de l'ordre de $1 + 2 + \dots + (n - 1)$, donc de l'ordre de n^2.

> **RÉPONSE À RETENIR**
>
> **Tri par insertion ; coût temporel quadratique** dans le pire des cas.

Question 9.

Sujet p. 6, 7

> Recopier et compléter la fonction charge(tab: list, volume: int) -> list afin qu'elle applique l'algorithme ChargementGlouton décrit plus haut.

On commence par trier les marchandises par ratio décroissant. On les parcourt ensuite dans cet ordre : si une marchandise tient dans le volume restant, on l'ajoute au chargement et on met à jour le volume restant.

```
def charge(tab: list, volume: int) -> list:
    tri(tab)
    chargement = []
    v_restant = volume
    n = len(tab)
    for i in range(n):
        if tab[i].volume <= v_restant:
            chargement.append(tab[i])
            v_restant = v_restant - tab[i].volume
    return chargement
```

Avec les données de la partie B et une capacité de 100 litres, le tri donne [m_3, m_2, m_1, m_4]. On prend m_3, ce qui laisse 60 litres. m_2, de volume 70 litres, ne tient pas ; on prend m_1, ce qui laisse 40 litres. m_4 ne tient pas dans le volume restant. Le chargement est donc [m_3, m_1], de volume 60 litres et de prix 160 + 40 = 200 €.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Avec une capacité de 100 litres et les marchandises de la partie B, la fonction renvoie **[m_3, m_1]**, pour un prix total de **200 €**.

Partie C – Rangement optimisé par récursivité

Question 10.

Sujet p. 7, 8

Compléter le code de la fonction récursive chargeOptimale(tab: list, v_restant: int, i: int), dont le principe a été décrit précédemment.

Si `i` atteint la taille du tableau, toutes les marchandises ont été examinées et la liste vide est le seul chargement restant. Si la marchandise d'indice `i` ne tient pas, on passe à la suivante sans modifier le volume restant. Sinon, on compare les deux possibilités : ne pas prendre cette marchandise, ou la prendre et poursuivre avec le volume restant diminué. Chaque appel récursif augmente `i` ; la fonction atteint donc son cas de base après au plus `len(tab) - i` étapes.

```
def chargeOptimale(tab: list, v_restant: int, i: int) -> list:
    if i >= len(tab):
        return []
    else:
        if tab[i].volume > v_restant:
            return chargeOptimale(tab, v_restant, i + 1)
        else:
            option1 = chargeOptimale(tab, v_restant, i + 1)
            option2 = [tab[i]] + chargeOptimale(
                tab, v_restant - tab[i].volume, i + 1
            )
            if prixListe(option1) > prixListe(option2):
                return option1
            else:
                return option2
```

Le code compare les deux choix pour la marchandise courante et conserve celui dont le prix total est le plus élevé. Avec [m_1, m_2, m_3, m_4] et 100 litres, la branche qui prend m_1 peut ensuite prendre m_2 : le chargement [m_1, m_2] vaut 250 €. Si m_1 n'est pas prise, le meilleur prix possible est 210 € : on peut obtenir ce prix avec m_2 seule ou avec m_3 et m_4. La fonction renvoie donc une combinaison de prix maximal, [m_1, m_2], pour 250 €. En cas d'égalité, le code renvoie `option2`, qui est également optimale. Dans le pire des cas, l'exploration des choix comporte un nombre d'appels de l'ordre de 2^n, et la profondeur maximale de récursion est n.

RÉPONSE À RETENIR

La fonction compare les solutions avec et sans chaque marchandise et renvoie un chargement de prix maximal. Pour les données de la partie B et une capacité de 100 litres, une solution optimale est **[m_1, m_2]**, de prix **250 €**.

FormaV - document d'entraînement non officiel

Exercice 3

8 points

Programmation orientée objet, classes Piste et Domaine, et recherche de chemins dans un graphe

Partie A – Analyse des classes Piste et Domaine

Question 1.

Sujet p. 9, 14

> Lister les attributs de la classe Piste en précisant leur type.

Les attributs de `Piste` sont les valeurs affectées à `self` dans le constructeur. La classe ne comporte pas d'annotations de type : les types sont donc déduits des données et de leur usage. En particulier, `longueur` peut être un entier ou un flottant selon la valeur transmise.

RÉPONSE À RETENIR

- `nom` : chaîne de caractères (`str`) ;
- `denivele` : entier (`int`), en mètres ;
- `longueur` : nombre représentant une distance en kilomètres. Dans les données, certaines longueurs sont des flottants et Vincendières a pour longueur la valeur entière `3` ;
- `couleur` : chaîne de caractères (`str`) ;
- `ouverte` : booléen (`bool`).

Question 2.

Sujet p. 9, 14

> Écrire la méthode set_couleur de la classe Piste qui permet d'affecter à l'attribut couleur la chaîne de caractères correspondant à la couleur de la piste : noire si son dénivelé est supérieur ou égal à 100 mètres ; rouge si son dénivelé est strictement inférieur à 100 mètres et supérieur ou égal à 70 mètres ; bleue s'il est strictement inférieur à 70 mètres et supérieur ou égal à 40 mètres ; verte s'il est strictement inférieur à 40 mètres.

On teste les seuils du plus grand au plus petit. Les seuils sont inclusifs : un dénivelé de 100 mètres donne « noire », 70 mètres donne « rouge » et 40 mètres donne « bleue ». Tout dénivelé inférieur à 40 mètres donne « verte ».

RÉPONSE À RETENIR

```
def set_couleur(self):
    if self.denivele >= 100:
        self.couleur = 'noire'
    elif self.denivele >= 70:
        self.couleur = 'rouge'
    elif self.denivele >= 40:
        self.couleur = 'bleue'
    else:
        self.couleur = 'verte'
```

Question 3. Proposition A

Sujet p. 10, 14

> Indiquer, parmi les quatre propositions, le type de l'élément renvoyé par l'instruction Python

> lievre_blanc.get_pistes(). Proposition A : une chaîne de caractères.

`get_pistes()` renvoie l'attribut `self.pistes`, qui est une liste : l'objet renvoyé n'est donc pas une chaîne de caractères.

> **RÉPONSE À RETENIR**
>
> Proposition A : fausse.

Question 3. Proposition B

Sujet p. 10, 14

> Indiquer, parmi les quatre propositions, le type de l'élément renvoyé par l'instruction Python lievre_blanc.get_pistes(). Proposition B : un objet de type Piste.

`get_pistes()` renvoie toute la liste des pistes, et non un seul objet `Piste`.

> **RÉPONSE À RETENIR**
>
> Proposition B : fausse.

Question 3. Proposition C

Sujet p. 10, 14

> Indiquer, parmi les quatre propositions, le type de l'élément renvoyé par l'instruction Python lievre_blanc.get_pistes(). Proposition C : une liste de chaînes de caractères.

La méthode `ajouter_piste` ajoute à la liste un objet créé par `Piste(...)`, et non une chaîne de caractères correspondant au nom de la piste.

> **RÉPONSE À RETENIR**
>
> Proposition C : fausse.

Question 3. Proposition D

Sujet p. 10, 14

> Indiquer, parmi les quatre propositions, le type de l'élément renvoyé par l'instruction Python lievre_blanc.get_pistes(). Proposition D : une liste d'objets de type Piste.

`get_pistes()` renvoie `self.pistes`. Cette liste contient les objets créés par `Piste(...)` dans `ajouter_piste`.

> **RÉPONSE À RETENIR**
>
> Proposition D : vraie. C'est la bonne proposition.

Question 4.

Sujet p. 9, 10, 14, 15

> Écrire un programme Python dont l'exécution permet de procéder à la fermeture de toutes les pistes vertes en affectant la valeur False à l'attribut ouverte des pistes concernées.

Les couleurs étant attribuées par le programme juste avant cette question, on parcourt la liste et on ferme chaque piste dont la couleur est `'verte'`. Pour être autonome, le code ci-dessous réattribue aussi les couleurs à partir des dénivelés. D'après l'annexe, les pistes vertes sont Verneys, Esseillon et Hermine.

RÉPONSE À RETENIR

```
for piste in lievre_blanc.get_pistes():
    piste.set_couleur()
    if piste.get_couleur() == 'verte':
        piste.ouverte = False
```

Les pistes fermées sont Verneys, Esseillon et Hermine.

Question 5.

Sujet p. 10, 14, 15

Écrire une fonction pistes_de_couleur prenant pour paramètres une chaîne de caractères couleur représentant la difficulté d'une piste et une liste lst de pistes de ski de fond. Cette fonction renvoie la liste des noms des pistes dont couleur est le niveau de difficulté. Exemple : l'instruction pistes_de_couleur(lievre_blanc.get_pistes(), 'noire') renvoie la liste ['Petit Bonheur', 'Forêt', 'Duvallon'].

On parcourt la liste des pistes et on ajoute au résultat le nom de celles dont la couleur correspond à celle demandée. La formulation du sujet présente les paramètres dans l'ordre couleur puis liste, mais l'exemple d'appel les donne dans l'ordre liste puis couleur. La fonction ci-dessous suit l'ordre de l'exemple.

RÉPONSE À RETENIR

```
def pistes_de_couleur(lst, couleur):
    resultat = []
    for piste in lst:
        if piste.get_couleur() == couleur:
            resultat.append(piste.get_nom())
    return resultat
```

Les couleurs doivent avoir été attribuées au préalable. Avec les données de l'annexe, l'appel `pistes_de_couleur(lievre_blanc.get_pistes(), 'noire')` renvoie `['Petit Bonheur', 'Forêt', 'Duvallon']`.

Question 6.

Sujet p. 10, 11, 14, 15

Recopier et compléter la fonction semi_marathon(L). Elle prend en paramètre une liste L de noms de pistes et renvoie True si la distance totale parcourue est strictement supérieure à 21,1 kilomètres, False sinon.

On initialise la distance à zéro, puis on ajoute la longueur de chaque piste dont le nom correspond à un élément de L. Le test final est strictement supérieur à 21,1, conformément à la définition de la fonction dans le sujet.

Il existe une différence d'accent entre l'exemple de la page 11 (« Chateau enneigé ») et le nom de l'annexe (« Château enneigé »). La comparaison ci-dessous ignore cette différence sur â, afin que l'exemple fourni corresponde aux données de l'annexe.

RÉPONSE À RETENIR

```
def semi_marathon(L):
    distance = 0
    liste_pistes = lievre_blanc.get_pistes()
    for nom in L:
        for piste in liste_pistes:
            if piste.get_nom().replace('â', 'a') == nom.replace('â', 'a'):
                distance = distance + piste.get_longueur()
    return distance > 21.1
```

Pour `entrainement1`, le total est 2,5 + 10,7 + 5,5 + 6,8 = 25,5 km : la fonction renvoie `True`. Pour `entrainement2`, le total est 1,8 + 10,1 + 3,4 = 15,3 km : elle renvoie `False`.

Partie B – Recherche par force brute

Question 7.

Sujet p. 11

Écrire une instruction Python permettant d'afficher la longueur de la piste allant du sommet 'E' au sommet 'F'.

Dans le dictionnaire de dictionnaires, la longueur de l'arête entre deux sommets s'obtient par deux accès successifs : le sommet de départ, puis le sommet d'arrivée.

RÉPONSE À RETENIR

```
print(domaine['E']['F'])
```

L'instruction affiche `10.1` kilomètres.

Question 8.

Sujet p. 11, 12

Écrire une fonction voisins qui prend en paramètres un graphe G et un sommet s du graphe G et qui renvoie la liste des voisins du sommet s. Exemple : l'instruction voisins(domaine, 'B') renvoie la liste ['C', 'D', 'E', 'F'].

Les voisins de `s` sont les clés du dictionnaire associé à ce sommet. On les convertit en liste pour obtenir le type demandé.

RÉPONSE À RETENIR

```
def voisins(G, s):
    return list(G[s].keys())
```

Par exemple, `voisins(domaine, 'B')` renvoie `['C', 'D', 'E', 'F']`.

Question 9.

Sujet p. 11, 12

> Recopier et compléter la fonction longueur_chemin donnée ci-dessous. Elle prend en paramètres un graphe G et un chemin du graphe G sous la forme d'une liste de sommets et renvoie sa longueur en kilomètres. Exemple : l'instruction longueur_chemin(domaine, ['B', 'E', 'F', 'H']) renvoie le nombre flottant 15.3.

La longueur d'un chemin est la somme des poids des arêtes successives. À chaque itération, on ajoute le poids entre le sommet précédent et le sommet courant, puis on actualise le sommet précédent. Un chemin vide ou réduit à un seul sommet a une longueur nulle.

RÉPONSE À RETENIR

```
def longueur_chemin(G, chemin):
    precedent = None
    longueur = 0
    for i in range(1, len(chemin)):
        precedent = chemin[i - 1]
        longueur = longueur + G[precedent][chemin[i]]
    return longueur
```

Pour `['B', 'E', 'F', 'H']`, la longueur vaut 1,8 + 10,1 + 3,4 = 15,3 km.

Question 10.

Sujet p. 12

> Expliquer en quoi la fonction parcours est une fonction récursive.

Une fonction est récursive lorsqu'elle s'appelle elle-même. Dans `parcours`, l'appel `parcours(G, sommet, chemin + [sommet])` rappelle la même fonction pour explorer un chemin prolongé.

La récursion se termine : un sommet ne peut être ajouté que s'il ne figure pas déjà dans le chemin. Dans un graphe fini, un chemin ainsi construit contient au plus une fois chaque sommet et ne peut donc pas être prolongé indéfiniment.

RÉPONSE À RETENIR

`parcours` est récursive parce qu'elle s'appelle elle-même afin d'explorer les chemins prolongés.

Question 11.

Sujet p. 12, 13

> Recopier et compléter la fonction parcours_dep_arr ci-après, qui renvoie la liste des chemins partant du sommet depart et se terminant par le sommet arrivee dans le graphe G entré en paramètre. La liste renvoyée ne doit pas comporter de doublons. Plusieurs lignes de code sont nécessaires.

On filtre les chemins renvoyés pour ne conserver que ceux qui partent de `depart` et finissent à `arrivee`, puis on élimine les doublons. Comme la fonction `parcours` utilise une liste mutable par défaut, des chemins d'appels précédents peuvent aussi être présents : on vérifie donc que chaque arête du chemin appartient bien au graphe donné.

Si les deux sommets sont identiques, le chemin simple réduit à ce sommet est renvoyé.

RÉPONSE À RETENIR

```
def parcours_dep_arr(G, depart, arrivee):
    if depart == arrivee:
        return [[depart]]
    liste = parcours(G, depart)
    resultat = []
    for chemin in liste:
        if chemin[0] == depart and chemin[-1] == arrivee:
            valide = True
            for i in range(1, len(chemin)):
                if chemin[i] not in G[chemin[i - 1]]:
                    valide = False
            if valide and chemin not in resultat:
                resultat.append(chemin)
    return resultat
```

Question 12.

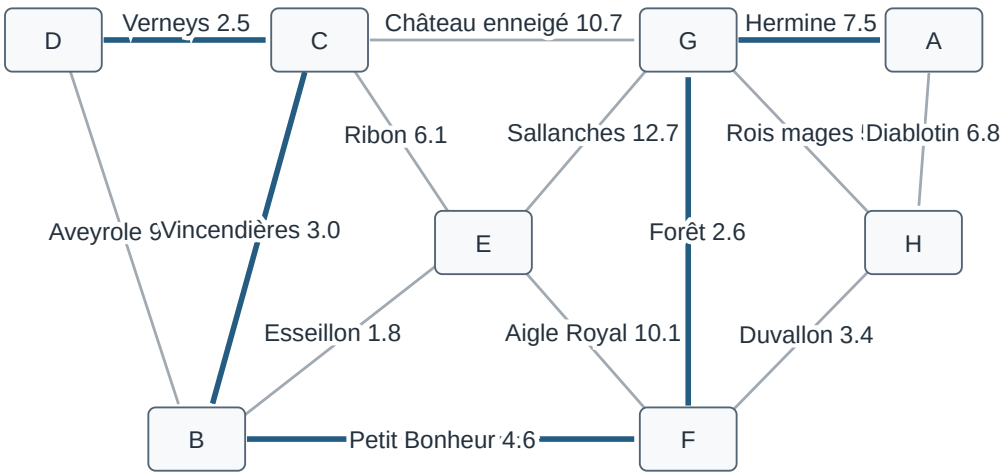
Sujet p. 11, 12, 13

Recopier et compléter la fonction plus_court. Elle prend pour paramètres un graphe G, un sommet de départ depart et un sommet d'arrivée arrivee ; elle renvoie un des chemins les plus courts sous la forme d'une liste de sommets.

On prend d'abord le premier chemin trouvé comme meilleur candidat, puis on compare sa longueur à celle de chaque autre chemin. Si aucun chemin n'existe, la fonction renvoie une liste vide. Les poids étant des longueurs positives, un plus court chemin peut être choisi sans cycle ; l'énumération des chemins simples suffit donc ici.

Pour aller de D à A, le chemin retenu est D → C → B → F → G → A. Sa longueur est 2,5 + 3,0 + 4,6 + 2,6 + 7,5 = 20,2 km.

Graphe pondéré du domaine et chemin de longueur minimale de D à A



Chaque arête non orientée est représentée une seule fois ; les poids sont en kilomètres. Le chemin mis en évidence est D → C → B → F → G → A.

RÉPONSE À RETENIR

```
def plus_court(G, depart, arrivee):
    liste_chemins = parcours_dep_arr(G, depart, arrivee)
    if liste_chemins == []:
        return []
    chemin_plus_court = liste_chemins[0]
    minimum = longueur_chemin(G, chemin_plus_court)
    for chemin in liste_chemins:
        longueur = longueur_chemin(G, chemin)
        if longueur < minimum:
            minimum = longueur
            chemin_plus_court = chemin
    return chemin_plus_court
```

Pour les secours, `plus_court(domaine, 'D', 'A')` renvoie `['D', 'C', 'B', 'F', 'G', 'A']`, de longueur 20,2 km.

Question 13.

Sujet p. 11, 13

> Expliquer en quoi le choix fait par le pisteur-secouriste de choisir la distance minimale pour arriver le plus rapidement possible sur le lieu de l'incident est discutable. Proposer un meilleur critère de choix.

La distance ne détermine pas toujours le temps de trajet : la pente, la difficulté et les conditions d'enneigement peuvent faire varier la vitesse. Un itinéraire plus long peut donc être parcouru plus rapidement qu'un itinéraire plus court.

Pour arriver le plus vite possible, il faudrait minimiser le temps estimé de parcours. On associerait à chaque arête une durée estimée, puis on choisirait le chemin dont la somme des durées est minimale. Les durées ou les vitesses n'étant pas fournies, on ne peut pas déterminer le trajet le plus rapide avec les seules données du sujet.

RÉPONSE À RETENIR

Le critère pertinent est le temps de parcours estimé, et non la seule distance. Il faudrait connaître une durée pour chaque piste afin de comparer les temps totaux.

Sujet : https://www.sujetdebac.fr/annales-pdf/2024/spe-numerique-informatique-2024-metropole-2-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.