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 (NSI)

Session 2023 • Jour 2

Code sujet **23-NSIJ2ME1** 3 h 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Bases de données et SQL
8 réponses • sujet p. 1–2

Exercice 2 — Réseaux, programmation et algorithmique
8 réponses • sujet p. 3–5

Exercice 3 — Arbres binaires de recherche
9 réponses • sujet p. 6–10

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

4 points

Bases de données et SQL

a

Question 1. (a)

Sujet p. 1, 2

> On n'oubliera pas de justifier les réponses pour les questions a. et b. Identifier pour chacune des deux relations présentées (prestations et clients) une clé primaire possible.

Une clé primaire identifie chaque ligne d'une relation sans ambiguïté. Dans les données fournies, les identifiants de prestation sont distincts, tout comme les identifiants client : chacun peut donc servir de clé primaire dans sa relation.

> **RÉPONSE À RETENIR**
>
> **prestations :** `id_prestation` est une clé primaire possible. **clients :** `id_client` est une clé primaire possible.

b

Question 1. (b)

Sujet p. 1, 2

> On n'oubliera pas de justifier les réponses pour les questions a. et b. Identifier la relation qui possède une clé étrangère. Donner cette clé étrangère.

Une clé étrangère est un attribut qui référence une clé primaire d'une autre relation. Les valeurs de `prestations.id_client` correspondent aux identifiants de clients présents dans `clients.id_client`.

> **RÉPONSE À RETENIR**
>
> La relation **prestations** possède la clé étrangère `id_client`, qui référence `clients(id_client)`.

c

Question 1. (c)

Sujet p. 1, 2

> Donner le type pour chacun des deux premiers attributs de la relation clients.

Le type se déduit de la nature des valeurs : l'identifiant client est un nombre entier et le nom est du texte.

> **RÉPONSE À RETENIR**
>
> `id_client` : entier. `nom` : chaîne de caractères (texte).

FormaV - document d'entraînement non officiel

a

Question 2. (a)

Sujet p. 1, 2

> On manipule les données en utilisant le langage SQL. Donner les valeurs renvoyées par la requête : SELECT nom, telephone FROM clients WHERE ville = 'Montélimar';

La requête conserve les lignes de `clients` dont la ville vaut « Montélimar », puis affiche uniquement les attributs `nom` et `telephone`. Les quatre clients concernés sont Ouellet, Foucault, Croteau et Rivard. Aucun ordre de résultat n'est imposé par la requête.

RÉPONSE À RETENIR

nom	telephone
Ouellet	0475016031
Foucault	0475918885
Croteau	0475460794
Rivard	0475339127

b

Question 2. (b)

Sujet p. 1, 2

> Écrire la requête permettant d'obtenir la date et l'heure de début des prestations de plus d'une heure, réalisées par Didier.

On sélectionne les attributs demandés, `date` et `heure_debut`. La condition `duree > 1` retient les prestations de plus d'une heure, et la condition sur `employe` ne conserve que celles réalisées par Didier.

```
SELECT date, heure_debut
FROM prestations
WHERE duree > 1 AND employe = 'Didier';
```

Dans les données fournies, les prestations qui satisfont ces conditions sont les identifiants 876272, 876277, 876279, 876935, 876939, 876941 et 876943.

RÉPONSE À RETENIR

date	heure_debut
15/06/2020	15:00
16/06/2020	14:00
17/06/2020	09:00
18/11/2020	09:00
18/11/2020	14:00
19/11/2020	09:00
19/11/2020	14:00

Question 3.

Sujet p. 1, 2

> Avec les informations présentées en page précédente, donner les valeurs renvoyées par la requête : SELECT DISTINCT nom FROM clients JOIN prestations ON clients.id_client = prestations.id_client WHERE prestations.employe = 'Sylvie'; Le mot-clé DISTINCT permet d'éviter les doublons dans le résultat de la requête.

La jointure associe chaque prestation à son client grâce à l'égalité des attributs `id_client`. On conserve ensuite les prestations réalisées par Sylvie et on affiche le nom du client. Les identifiants clients concernés sont 5231 (Foucault), 5229 (Rouze), 5230 (Bonenfant), 5234 (Rivard) et 5232 (Croteau). `DISTINCT` supprime les répétitions de noms. Aucun ordre de résultat n'est imposé par la requête.

RÉPONSE À RETENIR

nom
Foucault
Rouze
Bonenfant
Rivard
Croteau

a

Question 4. (a)

Sujet p. 1, 2

> Pour permettre au logiciel de préparer les factures, on souhaite ajouter le tarif horaire de chaque prestation dans ce jeu de données. Expliquer quel(s) problème(s) peuvent apparaître si l'on décide d'ajouter uniquement un attribut tarif_horaire à la relation prestations.

Si le tarif horaire dépend du type de prestation, ajouter `tarif_horaire` directement dans `prestations` oblige à répéter le même tarif pour chaque prestation de ce type. Cette redondance peut entraîner une incohérence si le tarif n'est pas modifié sur toutes les lignes concernées (anomalie de mise à jour). Elle

empêche aussi d'enregistrer le tarif d'un type tant qu'aucune prestation de ce type n'existe ; supprimer la dernière prestation de ce type ferait perdre son tarif.

Ces problèmes reposent sur l'hypothèse que le tarif est commun à un type donné. Si le tarif est propre à chaque prestation et ne dépend pas de son type, le conserver dans `prestations` peut être approprié.

> **RÉPONSE À RETENIR**
>
> Pour éviter la répétition du tarif et les anomalies associées, on peut enregistrer une seule fois le tarif horaire dans une relation distincte, identifiée par le type de prestation.

b

Question 4. (b)

Sujet p. 1, 2

> Proposer un schéma relationnel qui permet d'éviter ce(s) problème(s). On pourra donner la réponse sous la forme tarifs(premier attribut, deuxième attribut, ...), prestations(premier attribut, ...) et clients(premier attribut, ...). Indiquer explicitement pour chaque relation les attributs, la clé primaire et les éventuelles clés étrangères ; les types ne sont pas demandés.

On suppose que le tarif horaire est déterminé par le `type` de prestation. La relation `tarifs` associe chaque type à un tarif ; l'attribut `type` de `prestations` référence alors ce type. La relation `clients` conserve ses attributs et sa clé primaire.

> **RÉPONSE À RETENIR**
>
> - **tarifs**(`type`, `tarif_horaire`) : clé primaire `type`.
> - **prestations**(`id_prestation`, `id_client`, `date`, `heure_debut`, `duree`, `type`, `employe`) : clé primaire `id_prestation` ; clés étrangères `id_client` vers `clients(id_client)` et `type` vers `tarifs(type)`.
> - **clients**(`id_client`, `nom`, `adresse`, `code_postal`, `ville`, `telephone`) : clé primaire `id_client`.

Exercice 2

4 points

Réseaux, programmation et algorithmique

Question 1.

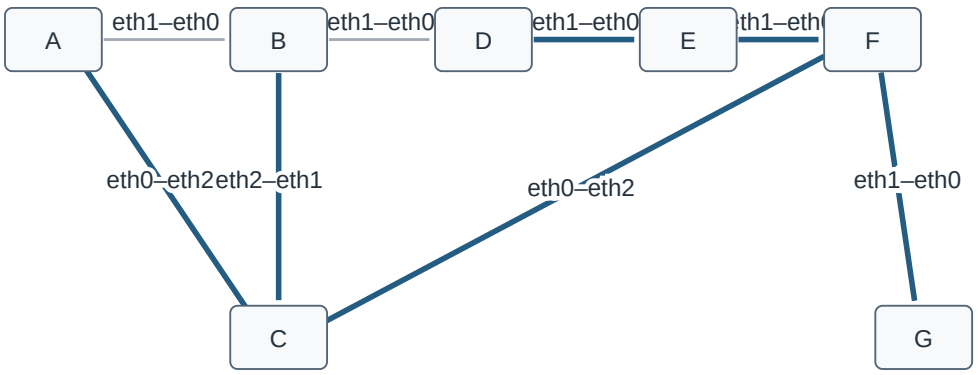
Sujet p. 3

> Pour vendre ses produits, un grossiste en confiseries utilise un service web. Son réseau local est représenté par un schéma dans lequel les nœuds A, B, C, D, E, F et G sont des routeurs. Selon le protocole RIP, la distance est le nombre de routeurs traversés par un chemin pour aller d'un routeur à un autre, et le chemin choisi minimise cette distance. Reproduire et compléter la table indiquant, pour chaque routeur A à F, la portion de la table de routage vers le routeur G : passerelle, interface et distance. La ligne A vers G est déjà renseignée : passerelle C, interface eth0, distance 2.

Les liaisons du schéma sont A–B, A–C, B–C, B–D, D–E, E–F, C–F et F–G. Pour déterminer une route vers G, on choisit à chaque routeur un voisin conduisant à un chemin de distance minimale. La ligne A déjà remplie donne le chemin A–C–F–G et une distance de 2 : conformément à cette valeur, la distance compte les routeurs intermédiaires, sans compter le routeur de départ ni G.

Les routes minimales sont donc, selon le routeur de départ : A–C–F–G, B–C–F–G, C–F–G, D–E–F–G, E–F–G et F–G. Les interfaces sont celles indiquées sur le schéma.

Réseau des routeurs et routes minimales vers G



Les liaisons mises en évidence appartiennent aux routes minimales vers G. Les étiquettes indiquent les interfaces aux deux extrémités.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Routeur	Destination	Passerelle	Interface	Distance
A	G	C	eth0	2
B	G	C	eth2	2
C	G	F	eth0	1
D	G	E	eth1	2
E	G	F	eth1	1
F	G	G	eth1	0

Par exemple, depuis B, la route minimale passe par C puis F avant d'atteindre G : elle compte deux routeurs intermédiaires. Depuis F, la liaison vers G est directe et ne comporte aucun routeur intermédiaire.

Question 2.

Sujet p. 3, 4

Pour écouler ses stocks, le grossiste applique une réduction de 10 % si le montant de la commande est supérieur ou égal à 100 € et strictement inférieur à 200 €, et une réduction de 20 % si le montant est supérieur ou égal à 200 €. Écrire en les complétant les lignes de code à partir de la ligne 13 de la fonction calcul_montant afin de respecter sa spécification : renvoyer le montant de la commande après réduction éventuelle.

Le montant avant réduction se calcule en multipliant le prix unitaire par la quantité pour chaque assortiment. Or la ligne 12 imprimée dans le sujet additionne les prix et les quantités : elle ne calcule pas le montant de la commande. Il faut donc corriger cette ligne pour que la fonction respecte sa spécification, puis compléter les lignes 13 à 17.

La première condition inclut 100 € et exclut 200 €. La seconde traite les montants à partir de 200 €. Si aucune condition n'est satisfaite, le montant n'est pas modifié.

RÉPONSE À RETENIR

```
def calcul_montant(prix_TF, quantite_TF, prix_TC, quantite_TC):
    montant = quantite_TF * prix_TF + quantite_TC * prix_TC
    if 100 <= montant < 200:
        montant = montant - montant * 0.10
    elif montant >= 200:
        montant = montant - montant * 0.20
    return montant
```

Pour la commande n° 343, le montant avant réduction vaut 1 × 7 + 3 × 16 = 55 €. Il est inférieur à 100 €, donc aucune réduction ne s'applique et la fonction renvoie 55 €.

a

Question 3. (a)

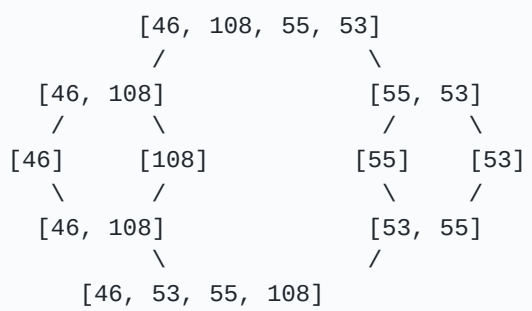
Sujet p. 4

> Le grossiste enregistre dans la liste sommes les montants des commandes avant réduction. Le tri-fusion trie la liste dans l'ordre croissant ; on suppose que le nombre d'éléments de la liste est une puissance de 2. Appliquer sur la copie, de la même façon que dans l'exemple, le déroulement de l'algorithme pour la liste [46, 108, 55, 53]. On rappelle que L.append(x) ajoute l'élément x à la fin de la liste L et que len(L) renvoie la taille de L.

On sépare la liste en deux moitiés, puis chacune de ces moitiés en listes d'un seul élément. On fusionne ensuite les listes triées en plaçant à chaque étape le plus petit élément disponible en premier.

Les deux premières fusions donnent [46, 108] et [53, 55]. La fusion finale compare leurs éléments dans l'ordre croissant.

RÉPONSE À RETENIR



La liste triée dans l'ordre croissant est [46, 53, 55, 108].

b

Question 3. (b) – Proposition 1

Sujet p. 5

> La fonction fusion renvoie une liste de valeurs triées à partir de deux listes liste1 et liste2 préalablement triées. Parmi les quatre propositions, écrire sur la copie l'instruction à placer à la ligne 4 du code. Proposition 1 : while i1 < len(liste1) and i2 < len(liste2):

Le corps de la boucle compare `liste1[i1]` et `liste2[i2]`. Ces deux accès sont valides tant que chacun des indices est inférieur à la taille de sa liste. La boucle de comparaison doit donc continuer uniquement lorsque les deux listes ont encore des éléments. Les boucles suivantes ajoutent ensuite les éléments restants de la liste non épuisée.

RÉPONSE À RETENIR

La proposition 1 est correcte :

```
while i1 < len(liste1) and i2 < len(liste2):
```

Question 3. (b) – Proposition 2

Sujet p. 5

> La fonction fusion renvoie une liste de valeurs triées à partir de deux listes liste1 et liste2 préalablement triées. Parmi les quatre propositions, écrire sur la copie l'instruction à placer à la ligne 4 du code. Proposition 2 : while i1 < len(liste1) or i2 < len(liste2):

Avec `or`, la boucle continuerait dès qu'une seule des deux listes a encore des éléments. Pourtant, son corps compare toujours un élément de chacune des listes. Si l'une est épuisée, l'accès à son indice provoque une erreur.

> **RÉPONSE À RETENIR**
>
> La proposition 2 est incorrecte. Il faut vérifier que les deux indices sont valides ; la bonne condition utilise `and`.

Question 3. (b) – Proposition 3

Sujet p. 5

> La fonction fusion renvoie une liste de valeurs triées à partir de deux listes liste1 et liste2 préalablement triées. Parmi les quatre propositions, écrire sur la copie l'instruction à placer à la ligne 4 du code. Proposition 3 : while i1 + i2 < len(liste1) + len(liste2):

La somme des indices ne garantit pas que chacun d'eux est valide. Par exemple, avec `liste1 = [1]` et `liste2 = [2]`, après avoir consommé le premier élément de `liste1`, on a `i1 = 1` et `i2 = 0`. La somme 1 est encore strictement inférieure à la somme des longueurs 2, mais `liste1[1]` n'existe pas : le corps de la boucle provoquerait une erreur.

> **RÉPONSE À RETENIR**
>
> La proposition 3 est incorrecte : il faut vérifier séparément que chaque indice est strictement inférieur à la longueur de sa liste.

Question 3. (b) – Proposition 4

Sujet p. 5

> La fonction fusion renvoie une liste de valeurs triées à partir de deux listes liste1 et liste2 préalablement triées. Parmi les quatre propositions, écrire sur la copie l'instruction à placer à la ligne 4 du code. Proposition 4 : while liste1[i1 + 1] < liste2[i2]):

Cette proposition utilise `i1 + 1` au lieu de l'indice courant `i1` et ne garantit pas que cet indice existe. De plus, telle qu'elle est imprimée dans le sujet, elle comporte une parenthèse fermante supplémentaire et n'est pas une instruction Python syntaxiquement valide.

> **RÉPONSE À RETENIR**
>
> La proposition 4 est incorrecte. La condition adaptée est celle de la proposition 1.

c

Question 3. (c)

Sujet p. 5

> Recopier et compléter la fonction récursive tri_fusion, qui prend en paramètre une liste non triée et la renvoie sous la forme d'une nouvelle liste triée. Utiliser la fonction fusion de la question précédente. Exemple : tri_fusion([140, 115, 78, 94]) doit renvoyer [78, 94, 115, 140]. Aide Python : si L est une liste, l'instruction L[i:j] renvoie la liste

FormaV - document d'entraînement non officiel

> constituée des éléments de L indexés de i à j−1 ; par exemple, si L = [5, 4, 8, 7, 3], alors L[2:4] vaut [8, 7].

Si la liste contient au plus un élément, elle est déjà triée. Sinon, on la coupe en deux à l'indice milieu, on trie récursivement chaque moitié, puis on fusionne les deux listes triées. À chaque appel récursif non terminal, la taille des listes diminue ; on atteint donc le cas de base.

Avec les données de la question, [46, 108, 55, 53] est partagée en [46, 108] et [55, 53]. Le tri récursif donne [46, 108] et [53, 55], dont la fusion donne [46, 53, 55, 108]. L'exemple du sujet se traite de la même manière : [140, 115, 78, 94] donne [115, 140] et [78, 94], puis [78, 94, 115, 140].

RÉPONSE À RETENIR

```
def tri_fusion(liste):
    if len(liste) <= 1:
        return liste
    milieu = len(liste) // 2
    liste1 = tri_fusion(liste[:milieu])
    liste2 = tri_fusion(liste[milieu:])
    return fusion(liste1, liste2)
```

Ainsi, `tri_fusion([46, 108, 55, 53])` renvoie `[46, 53, 55, 108]`.

Exercice 3

5 points

Arbres binaires de recherche

Question 1.

Sujet p. 6

> Parmi les trois arbres dessinés, recopier sur la copie le numéro correspondant à celui qui n'est pas un arbre binaire de recherche. Justifier. Dans un arbre binaire de recherche, les valeurs du sous-arbre gauche d'un nœud sont strictement inférieures à sa valeur et celles du sous-arbre droit lui sont supérieures ou égales.

Dans un arbre binaire de recherche, les valeurs doivent respecter les règles de classement par rapport à tous les ancêtres. Dans l'arbre 3, le nœud 20 se trouve dans le sous-arbre droit de 29. Il devrait donc être supérieur ou égal à 29, mais 20 < 29 : la propriété est violée.

> **RÉPONSE À RETENIR**
>
> **L'arbre qui n'est pas un arbre binaire de recherche est l'arbre 3.**

a

Question 2. (a)

Sujet p. 7

> En utilisant les méthodes de la classe ABR, écrire l'instruction Python qui permet d'instancier un objet a0, de type ABR, ayant un seul nœud (la racine) de valeur 18 ; écrire ensuite une séquence d'instructions qui permet d'insérer dans l'objet a0 les deux feuilles de valeurs 12 et 36.

Le constructeur reçoit la valeur de la racine, puis les sous-arbres gauche et droit. Comme un sous-arbre vide est représenté par `None`, la création de la racine seule nécessite deux valeurs `None`. La méthode `inserer_noeud` renvoyant un nouvel ABR, on réaffecte son résultat à `a0` après chaque insertion.

> **RÉPONSE À RETENIR**
>
> ```
> a0 = ABR(18, None, None)
> a0 = a0.inserer_noeud(12)
> a0 = a0.inserer_noeud(36)
> ```
>
> La valeur 12 est insérée à gauche de 18 car 12 < 18 ; la valeur 36 est insérée à droite car 36 > 18.

b

Question 2. (b)

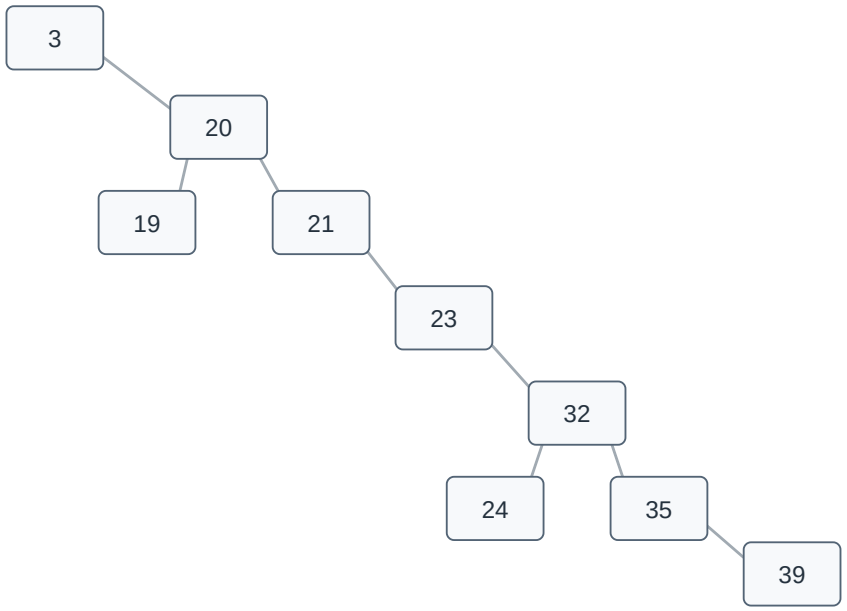
Sujet p. 7

> Dessiner sur la copie l'ABR nommé a2 obtenu en créant une instance de type ABR ayant un seul nœud (la racine) de valeur 3, puis en insérant successivement les valeurs dans l'ordre suivant : 20 ; 19 ; 21 ; 23 ; 32 ; 24 ; 35 ; 39.

En partant de 3, la valeur 20 est insérée à droite. La valeur 19, inférieure à 20, devient son fils gauche ; 21 devient son fils droit. Les valeurs 23, 32, 35 et 39 prolongent ensuite la branche droite. Enfin, 24, inférieur à 32 mais supérieur à 23, devient le fils gauche de 32.

FormaV - document d'entraînement non officiel

ABR a2 après les insertions



Les arêtes relient chaque nœud à son fils gauche ou droit.

RÉPONSE À RETENIR

L'ABR obtenu est :

```
3
 \
  20
 /  \
19   21
       \
        23
          \
           32
          /  \
         24   35
                \
                 39
```

c

Question 2. (c)

Sujet p. 6, 7

> Donner la hauteur des ABR a1 et a2. La hauteur d'un arbre ne contenant qu'un nœud vaut 1.

La hauteur est le nombre de nœuds sur le plus long chemin de la racine à une feuille. Dans a1, un chemin de hauteur maximale est 24 → 21 → 19 → 3 (ou 20), soit 4 nœuds. Dans a2, un chemin maximal est 3 → 20 → 21 → 23 → 32 → 35 → 39, soit 7 nœuds.

RÉPONSE À RETENIR

Hauteur de a1 : 4. Hauteur de a2 : 7.

d

Question 2. (d)

Sujet p. 6, 8

On complète la classe ABR avec une méthode calculer_hauteur qui renvoie la hauteur de l'arbre. Recopier sur la copie les lignes 10 et 13 en les complétant par des commentaires et la ligne 14 en la complétant par une instruction dans le code de cette méthode. On pourra utiliser la fonction Python max, qui prend en paramètres deux nombres et renvoie le maximum de ces deux nombres.

Si l'arbre n'a qu'un sous-arbre droit, sa hauteur est égale à 1 plus celle de ce sous-arbre. Si les deux sous-arbres existent, le plus long chemin passe par celui de plus grande hauteur : on ajoute donc 1 au maximum de leurs hauteurs. Chaque appel récursif porte sur un sous-arbre plus petit, jusqu'à atteindre une feuille.

RÉPONSE À RETENIR

```
elif self.sa_gauche is None:
    # arbre avec une racine et seulement un sous-arbre droit
    return 1 + self.sa_droit.calculer_hauteur()
else:
    # arbre avec une racine et deux sous-arbres
    return 1 + max(self.sa_gauche.calculer_hauteur(),
                   self.sa_droit.calculer_hauteur())
```

a

Question 3. (a)

Sujet p. 6, 7, 8

Recopier et compléter les lignes 6, 8, 11 et 13 du code de la méthode rechercher_valeur, qui permet de tester la présence ou l'absence d'une valeur donnée dans l'ABR.

On compare d'abord la valeur recherchée à celle du nœud courant. Si elles sont égales, la recherche réussit. Sinon, la propriété d'ABR permet de poursuivre uniquement dans le sous-arbre gauche si la valeur cherchée est plus petite, ou dans le sous-arbre droit si elle est plus grande. Si le sous-arbre correspondant n'existe pas, la valeur est absente.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
def rechercher_valeur(self, v):
    '''
    Renvoie True si la valeur v est trouvée dans l'ABR
    Renvoie False sinon
    '''
    if v == self.valeur:
        return True
    elif v < self.valeur and self.sa_gauche is not None:
        return self.sa_gauche.rechercher_valeur(v)
    elif v > self.valeur and self.sa_droit is not None:
        return self.sa_droit.rechercher_valeur(v)
    else:
        return False
```

b

Question 3. (b)

Sujet p. 7, 8

> On admet que le nombre de fois où la méthode rechercher_valeur est appelée pour rechercher la valeur 39 dans l'ABR a2 est 7. Donner le nombre de fois où la méthode rechercher_valeur est appelée pour rechercher la valeur 20 dans l'ABR a1.

Dans a1, la recherche de 20 suit le chemin 24 → 21 → 19 → 20. La méthode est donc appelée une fois à chacun de ces quatre nœuds, puis la recherche s'arrête lorsque 20 est trouvé.

RÉPONSE À RETENIR

La méthode est appelée 4 fois.

a

Question 4. (a)

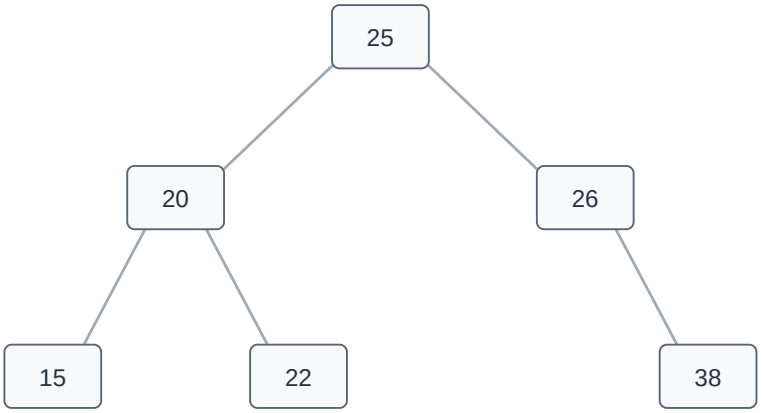
Sujet p. 9, 10

> En suivant les différentes étapes de l'algorithme de rotation gauche décrit dans l'énoncé, dessiner l'arbre obtenu après une rotation gauche de l'ABR représenté : racine 20, fils gauche 15, fils droit 25 ; le nœud 25 a pour fils gauche 22 et fils droit 26 ; le nœud 26 a pour fils droit 38.

Le pivot d'une rotation gauche est le fils droit de la racine : ici, le nœud 25. Son sous-arbre gauche, enraciné en 22, devient le sous-arbre droit de l'ancienne racine 20. Cette ancienne racine devient ensuite le fils gauche de 25. Les autres liens sont conservés : 15 reste le fils gauche de 20, 26 reste le fils droit de 25 et 38 reste le fils droit de 26.

FormaV - document d'entraînement non officiel

Arbre après la rotation gauche



Le nœud 22, ancien fils gauche du pivot 25, devient le fils droit de 20.

RÉPONSE À RETENIR

Après la rotation gauche, l'arbre est :

```
      25
     /  \
   20    26
  /  \     \
 15  22     38
```

b

Question 4. (b)

Sujet p. 9

> Écrire le code d'une méthode Python rotation_gauche qui réalise la rotation gauche d'un ABR autour de sa racine.

On applique les étapes symétriques de la rotation droite de l'énoncé. Le pivot est le sous-arbre droit. Son sous-arbre gauche devient le sous-arbre droit de la racine initiale, puis cette racine devient le sous-arbre gauche du pivot. Comme pour la méthode fournie pour la rotation droite, on suppose que le sous-arbre pivot existe.

RÉPONSE À RETENIR

```
def rotation_gauche(self):
    '''Renvoie une instance d'un ABR après une rotation gauche.
    On suppose qu'il existe un sous-arbre droit.
    '''
    pivot = self.sa_droit
    self.sa_droit = pivot.sa_gauche
    pivot.sa_gauche = self
    return ABR(pivot.valeur, pivot.sa_gauche, pivot.sa_droit)
```

Sujet : https://aeif.fr/wp-content/uploads/2025/07/Liban_mars_2023_2.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.