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 2022 • Jour 1

Code sujet **22-NSIJ1LR1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Structures de données : listes, piles et files
7 réponses • sujet p. 2–3

Exercice 2 — Programmation objet : gestion de bulles dans un jeu
4 réponses • sujet p. 4–6

Exercice 3 — Bases de données relationnelles et langage SQL
9 réponses • sujet p. 7–16

Exercice 4 — Arbres binaires : parcours en profondeur préfixe et infixe
9 réponses • sujet p. 10–12

Exercice 5 — Transmission de données dans un réseau, architecture d’un réseau, protocoles de routage, langages et programmation
15 réponses • sujet p. 13–15

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

4 points

Structures de données : listes, piles et files

1

Question 1.a

Sujet p. 2

> Les quatre questions suivantes sont indépendantes : pour chacune, repartir de la pile p et de la file f initiales présentées page 2. Représenter la file f après l'exécution de enfiler(f, defiler(f)).

La tête de la file est à droite. `defiler(f)` retire donc 1. Puis `enfiler(f, 1)` ajoute 1 à la queue, située à gauche.

> **RÉPONSE À RETENIR**
>
> File de gauche à droite : **1, 4, 3, 8, 2**.

Question 1.b

Sujet p. 2

> Les quatre questions suivantes sont indépendantes : pour chacune, repartir de la pile p et de la file f initiales présentées page 2. Représenter la pile p après l'exécution de empiler(p, depiler(p)).

Le sommet de la pile est 5. `depiler(p)` retire 5, puis `empiler(p, 5)` le remet au sommet. Les autres éléments ne changent pas.

> **RÉPONSE À RETENIR**
>
> Pile de haut en bas : **5, 8, 6, 2**.

Question 1.c

Sujet p. 2

> Les quatre questions suivantes sont indépendantes : pour chacune, repartir de la pile p et de la file f initiales présentées page 2. Représenter la pile p et la file f après l'exécution du code : for i in range(2): enfiler(f, depiler(p)).

La pile initiale a pour sommet 5. Les deux dépilages retirent successivement 5 puis 8. Chacun est ajouté à la queue de la file, à gauche : 5 est ajouté en premier, puis 8 devant lui. La pile perd ces deux éléments.

> **RÉPONSE À RETENIR**
>
> File de gauche à droite : **8, 5, 4, 3, 8, 2, 1**. Pile de haut en bas : **6, 2**.

Question 1.d

Sujet p. 2

> Les quatre questions suivantes sont indépendantes : pour chacune, repartir de la pile p et de la file f initiales présentées page 2. Représenter la pile p et la file f après l'exécution du code : for i in range(2): empiler(p, defiler(f)).

La tête de la file étant à droite, les deux appels à `defiler` retirent successivement 1 puis 2. Ils sont empilés dans cet ordre : 2 devient donc le sommet. La file perd ces deux éléments à droite.

> **RÉPONSE À RETENIR**
>
> File de gauche à droite : **4, 3, 8**. Pile de haut en bas : **2, 1, 5, 8, 6, 2**.

Question 2.

Sujet p. 3

> Préciser l'état de la variable f après chaque boucle de la fonction mystere appliquée à la file représentée dans le sujet. Indiquer le contenu de la pile renvoyée par la fonction.

La file donnée sur la page 3 est représentée de gauche à droite par 1, 2, 3, 4 ; sa tête est à droite, donc la première valeur retirée est 4. La première boucle retire ainsi 4, 3, 2, 1 et les empile dans cet ordre. À la fin, la file est vide et la pile a pour sommet 1, suivie de 2, 3, 4.

La deuxième boucle dépile 1, puis 2, puis 3, puis 4 et les ajoute successivement à la queue, à gauche. Après ces ajouts, l'ordre de gauche à droite est 4, 3, 2, 1. La pile a été entièrement dépilée.

> **RÉPONSE À RETENIR**
>
> Après la première boucle, **f est vide**. Après la deuxième boucle, **f vaut 4, 3, 2, 1** de gauche à droite. La pile renvoyée par la fonction est **vide**.

3

Question 3.a

Sujet p. 3

> Recopier et compléter le tableau qui détaille, étape par étape, le fonctionnement de l'algorithme knuth(f) pour la file 2, 1, 3. Une étape correspond à une modification de la pile ou de la file ; le nombre de colonnes peut être modifié.

On écrit les files de la queue (gauche) vers la tête (droite), et les piles du sommet vers le bas. Une étape correspond ici à une seule modification de la pile ou de la file : les dépilages et les enfilages sont donc présentés séparément.

Par exemple, après le retrait de 1, la condition de la boucle `while` est vraie car 1 est inférieur au sommet 3. On dépile alors 3, on l'ajoute à la queue, puis on empile 1. Les mêmes règles donnent les opérations des itérations suivantes et du dépilage final.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Modification	File f (queue → tête)	Pile p (sommet → bas)
État initial	2, 1, 3	vide
Retirer 3 de la tête	2, 1	vide
Empiler 3	2, 1	3
Retirer 1 de la tête	2	3
Dépiler 3	2	vide
Ajouter 3 à la queue	3, 2	vide
Empiler 1	3, 2	1
Retirer 2 de la tête	3	1
Empiler 2	3	2, 1
Dépiler 2	3	1
Ajouter 2 à la queue	2, 3	1
Dépiler 1	2, 3	vide
Ajouter 1 à la queue	1, 2, 3	vide

Question 3.b

Sujet p. 3

> Que fait cet algorithme ?

Pendant les `N` itérations, les éléments initiaux sont traités dans l'ordre où ils sont retirés de la tête. Les valeurs déplacées temporairement vers la queue ne passent pas devant les éléments initiaux qui restent à traiter, puisque la queue est à gauche et la tête à droite.

La pile conserve ses valeurs dans un ordre décroissant du sommet vers le bas : lorsqu'une nouvelle valeur est inférieure au sommet, les valeurs plus grandes sont dépilées et déplacées vers la queue avant que la nouvelle valeur soit empilée. À la fin des itérations, le dépilage de la pile fournit donc les valeurs de la plus grande à la plus petite. Les ajouter successivement à la queue inverse cet ordre dans la file.

RÉPONSE À RETENIR

L'algorithme trie la file en ordre croissant de la queue vers la tête (et donc en ordre décroissant si l'on retire successivement les éléments depuis la tête). Pour l'exemple, la file finale est 1, 2, 3 de gauche à droite.

FormaV - document d'entraînement non officiel

Exercice 2

4 points

Programmation objet : gestion de bulles dans un jeu

1

Question 1.a

Sujet p. 5

> Recopier les quatre dernières lignes et compléter les blancs du code Python de donnePremierIndiceLibre(Mousse), afin que la fonction renvoie l'indice du premier emplacement libre contenant None dans la liste Mousse, ou renvoie 6 en l'absence d'un emplacement libre.

La liste `Mousse` contient six éléments, aux indices 0 à 5. On augmente `i` tant que l'on n'a pas dépassé le dernier indice et que l'emplacement courant n'est pas `None`. La condition `i < 6` est évaluée avant l'accès à `Mousse[i]` : on ne tente donc jamais de lire l'indice 6.

```
def donnePremierIndiceLibre(Mousse):
    """
    Mousse est une liste.
    La fonction doit renvoyer l'indice du premier
    emplacement libre (contenant None) dans la liste Mousse
    ou renvoyer 6 en l'absence d'un emplacement libre dans
    Mousse.
    """
    i = 0
    while i < 6 and Mousse[i] != None:
        i = i + 1
    return i
```

Si tous les emplacements sont occupés, la boucle s'arrête avec `i == 6`, valeur demandée par l'énoncé.

RÉPONSE À RETENIR

Pour `Mousse = [None, None, None, None, None, None]`, la fonction renvoie `0`. Si les six emplacements sont occupés, elle renvoie `6`.

Question 1.b

Sujet p. 5, 6

> Écrire la fonction placeBulle(B), qui reçoit en paramètre un objet de type Cbulle et place cet objet dans la liste Mousse à la place d'un None. La fonction ne renvoie rien et modifie la liste Mousse. Si aucun emplacement n'est disponible, elle ne modifie rien.

On recherche l'indice du premier emplacement libre. Si cet indice est inférieur à 6, on y place l'objet `B`. Si l'indice vaut 6, aucun emplacement n'est libre et la fonction ne modifie pas la liste. On suppose que `Mousse` est la liste déjà créée dans le programme, comme dans l'énoncé.

```
def placeBulle(B):
    indice = donnePremierIndiceLibre(Mousse)
    if indice < 6:
        Mousse[indice] = B
```

Il n'est pas nécessaire d'écrire `global Mousse` : la fonction modifie un élément de la liste, sans réaffecter la variable `Mousse`. En l'absence d'instruction `return`, Python renvoie implicitement `None`.

RÉPONSE À RETENIR

La bulle est placée dans le premier emplacement contenant `None`. Si aucun emplacement n'est libre, la liste reste inchangée.

Question 2.

Sujet p. 6

> Écrire la fonction bullesEnContact(B1, B2), qui renvoie True si la bulle B2 touche la bulle B1 et False dans le cas contraire. Deux bulles sont en contact si la distance qui sépare leurs centres est inférieure ou égale à la somme de leurs rayons. La fonction distanceEntreBulles(B1, B2), qui calcule cette distance, est disponible.

Les bulles sont en contact lorsque la distance entre leurs centres est inférieure ou égale à la somme de leurs rayons. La fonction `distanceEntreBulles` fournit cette distance ; il suffit donc de comparer sa valeur à la somme des deux rayons.

```
def bullesEnContact(B1, B2):
    return distanceEntreBulles(B1, B2) <= B1.rayon + B2.rayon
```

L'égalité est incluse : deux bulles tangentes sont en contact.

RÉPONSE À RETENIR

La fonction renvoie `True` si `distanceEntreBulles(B1, B2) <= B1.rayon + B2.rayon`, et `False` sinon.

Question 3.

Sujet p. 4, 6

> Recopier et compléter les blancs de la fonction collision(indPetite, indGrosse, mousse). Calculer le nouveau rayon de la grosse bulle à partir des surfaces, réduire de 50 % sa vitesse et supprimer la petite bulle dans Mousse.

La surface après absorption est la somme des surfaces des deux bulles. Comme la surface d'un disque de rayon `r` est `pi * r**2`, le nouveau rayon est la racine carrée de la surface obtenue divisée par `pi`. Il faut affecter ce rayon à la grosse bulle pour que sa taille soit effectivement mise à jour. Diviser chacune de ses deux composantes de déplacement par 2 réduit de moitié la norme de son vecteur vitesse. Enfin, l'emplacement de la petite bulle devient `None`.

Le paramètre de la fonction s'appelle `mousse` (avec une minuscule). Le code utilise ce nom partout : Python distingue les majuscules des minuscules. L'affectation du nouveau rayon est ajoutée après son calcul, car elle est nécessaire pour modifier effectivement l'objet.

```
def collision(indPetite, indGrosse, mousse):
    """
    Absorption de la plus petite bulle d'indice indPetite
    par la plus grosse bulle d'indice indGrosse. Aucun test
    n'est réalisé sur les positions.
    """
    # calcul du nouveau rayon de la grosse bulle
    surfPetite = pi * mousse[indPetite].rayon**2
    surfGrosse = pi * mousse[indGrosse].rayon**2
    surfGrosseApresCollision = surfPetite + surfGrosse
    rayonGrosseApresCollision = sqrt(surfGrosseApresCollision / pi)
    mousse[indGrosse].rayon = rayonGrosseApresCollision

    # réduction de 50 % de la vitesse de la grosse bulle
    mousse[indGrosse].dirx = mousse[indGrosse].dirx / 2
    mousse[indGrosse].diry = mousse[indGrosse].diry / 2

    # suppression de la petite bulle dans mousse
    mousse[indPetite] = None
```

Par exemple, si les surfaces initiales sont 1 et 4, la surface après collision vaut 5 et le rayon calculé est `sqrt(5 / pi)`. Le rayon obtenu par cette formule peut être non entier, même si le rayon initial est décrit comme un entier ; le calcul de surface fourni par l'énoncé conduit directement à cette valeur et aucune règle d'arrondi n'est indiquée.

> **RÉPONSE À RETENIR**
>
> La grosse bulle reçoit la somme des surfaces, son rayon est mis à jour avec `sqrt((surfPetite + surfGrosse) / pi)`, ses deux composantes de déplacement sont divisées par 2 et l'emplacement de la petite bulle dans `mousse` devient `None`.

Exercice 3

4 points

Bases de données relationnelles et langage SQL

1

Question 1.a

Sujet p. 7, 8

> Que retourne la requête suivante ? SELECT titre FROM `qcm` WHERE date>'2022-01-10' ;

La condition `date > '2022-01-10'` retient les QCM dont la date est strictement postérieure au 10 janvier 2022. Parmi les dates de la table `qcm`, le 8 avril 2022 et le 15 février 2022 satisfont cette condition. Le QCM « Arbre Binaire », daté du 9 janvier 2022, ne la satisfait pas.

RÉPONSE À RETENIR

La requête renvoie les titres suivants. Sans `ORDER BY`, leur ordre d'affichage n'est pas garanti.

titre
POO
Arbre Parcours

Question 1.b

Sujet p. 7, 8, 16

> Écrire une requête qui donne les notes de l'élève qui a pour identifiant 4.

Les notes sont dans la table `lien_eleve_qcm`. Il faut donc sélectionner `note` et ne retenir que les lignes où `ideleve` vaut 4.

```
SELECT note
FROM lien_eleve_qcm
WHERE ideleve = 4;
```

Dans les données fournies, les deux lignes de l'élève 4 portent les notes 15 et 20.

RÉPONSE À RETENIR

note
15
20

2

Question 2.a

Sujet p. 7, 8

> Sachant que la clé primaire de la relation lien_eleve_qcm est composée de l'association des deux attributs ideleve

FormaV - document d'entraînement non officiel

> et idqcm, expliquer pourquoi, avec ce schéma relationnel, un élève ne peut pas faire deux fois le même QCM.

Dans `lien_eleve_qcm`, la clé primaire est le couple `(ideleve, idqcm)`. Une clé primaire ne peut pas apparaître deux fois. Une seconde participation du même élève au même QCM créerait une ligne portant le même couple, ce qui est interdit par cette clé.

> **RÉPONSE À RETENIR**
>
> Avec ce schéma, un élève ne peut donc être enregistré qu'une seule fois pour un QCM donné.

Question 2.b

Sujet p. 7, 8

> L'élève Marty Mael vient de faire le QCM sur la POO et a obtenu une note de 18. Comment est/sont modifiée(s) le(s) table(s) ? Il n'est pas demandé d'écrire une requête SQL.

Dans `eleves`, Marty Mael a l'identifiant 4 ; dans `qcm`, le QCM « POO » a l'identifiant 2. Il faut donc enregistrer dans la table d'association ces deux identifiants avec la note obtenue. Le couple `(4, 2)` n'étant pas déjà présent dans les données fournies, il ne fait pas doublon avec la clé primaire composée.

> **RÉPONSE À RETENIR**
>
> On ajoute à `lien_eleve_qcm` la ligne `(ideleve, idqcm, note) = (4, 2, 18)`. Les tables `eleves` et `qcm` ne sont pas modifiées.

Question 2.c

Sujet p. 7, 8, 16

> Un nouvel élève (nom : Lefèvre, prenom : Kevin) est enregistré. Écrire la requête permettant la mise à jour du/des relation(s).

Le nouvel élève doit être ajouté à la table `eleves`. Le sujet ne précise pas si `ideleve` est généré automatiquement. Si la base le génère automatiquement, la requête suivante convient :

```
INSERT INTO eleves (nom, prenom)
VALUES ('Lefèvre', 'Kevin');
```

Si l'identifiant doit être fourni explicitement, on peut choisir 6, qui n'est pas utilisé dans les données fournies, à condition que cet identifiant soit bien disponible dans la base au moment de l'insertion :

```
INSERT INTO eleves (ideleve, nom, prenom)
VALUES (6, 'Lefèvre', 'Kevin');
```

> **RÉPONSE À RETENIR**
>
> Dans les deux cas, une ligne est ajoutée à `eleves`. Le choix entre ces deux écritures dépend de la génération automatique de `ideleve`, qui n'est pas indiquée dans le sujet.

Question 2.d

Sujet p. 7, 8, 16

> L'élève Dubois Thomas quitte l'établissement et toutes les références à cet élève doivent être supprimées des relations. Pour la relation lien_eleve_qcm, écrire la requête pour supprimer toutes les références à l'élève Dubois Thomas qui a pour identifiant 2.

Dans la table `lien_eleve_qcm`, toutes les lignes qui référencent Dubois Thomas ont `ideleve = 2`. On supprime ces lignes avec cette condition :

```
DELETE FROM lien_eleve_qcm
WHERE ideleve = 2;
```

RÉPONSE À RETENIR

Cette requête supprime de `lien_eleve_qcm` toutes les références à l'élève d'identifiant 2, sans supprimer les lignes concernant d'autres élèves.

3

Question 3.a

Sujet p. 7, 9

> Recopier et compléter les blancs de la requête afin qu'elle affiche la liste des noms et prénoms des élèves ayant fait le QCM d'idqcm égal à 4 : SELECT … FROM eleves JOIN lien_eleve_qcm ON eleves.ideleve = … WHERE … ;

On joint chaque élève à ses participations en faisant correspondre les attributs `ideleve`. On filtre ensuite sur l'identifiant 4 du QCM et on affiche le nom et le prénom :

```
SELECT eleves.nom, eleves.prenom
FROM eleves
JOIN lien_eleve_qcm ON eleves.ideleve = lien_eleve_qcm.ideleve
WHERE lien_eleve_qcm.idqcm = 4;
```

RÉPONSE À RETENIR

Dans les données fournies, les lignes de `lien_eleve_qcm` où `idqcm = 4` concernent les élèves 2, 4 et 5. La jointure renvoie donc :

nom	prenom
Dubois	Thomas
Marty	Mael
Bikila	Abebe

Question 3.b

Sujet p. 8, 9

> Donner le résultat de la requête de la question 3.a.

Les participations au QCM d'identifiant 4 sont celles des élèves 2, 4 et 5. En reliant ces identifiants à la table `eleves`, on obtient les trois noms et prénoms ci-dessous.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

nom	prenom
Dubois	Thomas
Marty	Mael
Bikila	Abebe

Question 4.

Sujet p. 7, 8, 9, 16

> Écrire une requête qui affiche le nom, le prénom et la note des élèves ayant fait le QCM Arbre Binaire. L'utilisation des trois tables dans la requête est attendue.

Il faut relier `eleves` à `lien_eleve_qcm` par `ideleve`, puis cette table à `qcm` par `idqcm`. On filtre enfin les lignes sur le titre « Arbre Binaire » et on sélectionne le nom, le prénom et la note :

```
SELECT eleves.nom, eleves.prenom, lien_eleve_qcm.note
FROM eleves
JOIN lien_eleve_qcm ON eleves.ideleve = lien_eleve_qcm.ideleve
JOIN qcm ON lien_eleve_qcm.idqcm = qcm.idqcm
WHERE qcm.titre = 'Arbre Binaire';
```

Dans `qcm`, « Arbre Binaire » a l'identifiant 3. Dans `lien_eleve_qcm`, les lignes de cet identifiant concernent les élèves 2 et 3, avec une note de 18 chacun.

RÉPONSE À RETENIR

La requête renvoie les lignes suivantes. Sans `ORDER BY`, leur ordre d'affichage n'est pas garanti.

nom	prenom	note
Dubois	Thomas	18
Dupont	Cassandra	18

Exercice 4

4 points

Arbres binaires : parcours en profondeur préfixe et infixe

1

Question 1.a

Sujet p. 10, 11

> Expliquer en quoi cet arbre généalogique est un arbre binaire.

Dans la représentation donnée, les enfants d'un nœud sont les parents de l'individu correspondant à ce nœud. Il y en a au plus deux : le père, placé dans le sous-arbre gauche, et la mère, dans le sous-arbre droit. Si les parents ne sont pas connus, les deux sous-arbres valent `None`.

> **RÉPONSE À RETENIR**
>
> C'est un arbre binaire car chaque nœud possède au plus deux sous-arbres : celui du père à gauche et celui de la mère à droite.

Question 1.b

Sujet p. 10, 11

> Pourquoi un arbre généalogique n'est-il pas un arbre binaire de recherche (ABR) ?

Un arbre binaire de recherche doit respecter une règle d'ordre : les valeurs du sous-arbre gauche sont inférieures à celle du nœud et celles du sous-arbre droit lui sont supérieures. Ici, les côtés gauche et droit indiquent respectivement le père et la mère ; ils ne correspondent pas à un classement des noms ou des prénoms.

> **RÉPONSE À RETENIR**
>
> Un arbre généalogique n'est pas nécessairement un ABR : ses sous-arbres représentent le père et la mère, et non des personnes ordonnées selon leurs noms ou leurs prénoms.

2

Question 2.a

Sujet p. 10, 11

> Écrire les sept premières personnes (nom et prénom) rencontrées si on utilise le parcours en profondeur préfixe.

Le parcours préfixe visite le nœud courant, puis son sous-arbre gauche, puis son sous-arbre droit. En partant d'Albert, les visites commencent par Albert, Jules (son père), Michel (le père de Jules), le Jules situé sous Michel, Odile, Hélène, puis Evariste.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Les sept premières personnes rencontrées sont :

1. Albert Normand
2. Jules Normand
3. Michel Normand
4. Jules Normand
5. Odile Picard
6. Hélène Breton
7. Evariste Breton

Question 2.b

Sujet p. 10, 11

> Écrire les sept premières personnes (nom et prénom) rencontrées si on utilise le parcours en profondeur infixe.

Le parcours infixe visite le sous-arbre gauche, puis le nœud courant, puis le sous-arbre droit. Dans le sous-arbre gauche d'Albert, on visite d'abord le Jules situé sous Michel, puis Michel et Odile ; on revient ensuite à Jules, avant de parcourir le sous-arbre gauche d'Hélène et de visiter Hélène puis Camélia.

RÉPONSE À RETENIR

Les sept premières personnes rencontrées sont :

1. Jules Normand
2. Michel Normand
3. Odile Picard
4. Jules Normand
5. Evariste Breton
6. Hélène Breton
7. Camélia Charentais

Question 2.c

Sujet p. 11

> Recopier et compléter l'algorithme de parcours en profondeur fourni en y insérant au bon endroit la ligne contenant l'instruction d'affichage, pour que cet algorithme corresponde à un parcours en profondeur préfixe.

En parcours préfixe, le nœud courant est affiché avant l'exploration de ses deux sous-arbres. Comme `identite` contient le tuple `(prenom, nom)`, on affiche ses éléments d'indice 0 puis 1.

RÉPONSE À RETENIR

```
def parcours(racine_de_l_arbre):
    if racine_de_l_arbre != None:
        noeud_actuel = racine_de_l_arbre
        print(noeud_actuel.identite[0], noeud_actuel.identite[1])
        parcours(noeud_actuel.gauche)
        parcours(noeud_actuel.droite)
```

Question 2.d

Sujet p. 11

> Recopier et compléter l'algorithme de parcours en profondeur fourni en y insérant au bon endroit la ligne contenant l'instruction d'affichage, pour que cet algorithme corresponde à un parcours en profondeur infixe.

En parcours infixe, on explore d'abord le sous-arbre gauche, puis on affiche le nœud courant, et enfin on explore le sous-arbre droit.

RÉPONSE À RETENIR

```
def parcours(racine_de_l_arbre):
    if racine_de_l_arbre != None:
        noeud_actuel = racine_de_l_arbre
        parcours(noeud_actuel.gauche)
        print(noeud_actuel.identite[0], noeud_actuel.identite[1])
        parcours(noeud_actuel.droite)
```

3

Question 3.a

Sujet p. 11

> Recopier et compléter la définition de la classe Noeud pour ajouter un attribut generation qui indique la génération d'un individu. Lors de la création de l'instance, sa valeur par défaut est 0.

L'attribut `generation` est initialisé dans le constructeur. Il vaut ainsi 0 par défaut pour chaque nouvel objet.

RÉPONSE À RETENIR

```
class Noeud():
    def __init__(self, prenom, nom):
        self.identite = (prenom, nom)
        self.gauche = None
        self.droite = None
        self.generation = 0
```

Question 3.b

Sujet p. 10, 12

> Écrire la fonction récursive numerotation qui parcourt l'arbre et modifie l'attribut generation de tous les ancêtres d'Albert Normand, de sorte que ses parents soient de génération 1, ses grands-parents de génération 2, etc. Cette fonction prend en paramètres racine_de_l_arbre de type Noeud et num_gen de type entier ; num_gen vaut 0 par défaut.

On attribue à chaque nœud la génération reçue en paramètre. Les appels récursifs vers le père et la mère augmentent ce numéro de 1. Ainsi, avec l'appel initial par défaut, Albert est en génération 0, ses parents en génération 1 et les générations suivantes sont numérotées de la même manière. Le test contre `None` arrête la récursion lorsqu'il n'y a plus de nœud.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
def numerotation(racine_de_l_arbre, num_gen=0):
    if racine_de_l_arbre != None:
        racine_de_l_arbre.generation = num_gen
        numerotation(racine_de_l_arbre.gauche, num_gen + 1)
        numerotation(racine_de_l_arbre.droite, num_gen + 1)
```

Question 4.

Sujet p. 10, 12

> Écrire, dans l'ordre d'affichage, le résultat de l'exécution de mystere(racine_de_l_arbre, False), où racine_de_l_arbre est le nœud qui référence Albert Normand.

La fonction affiche le prénom du nœud courant uniquement si `affiche` vaut `True`. Depuis l'appel initial avec `False`, Albert n'est pas affiché. Chaque appel au fils gauche reçoit `False` et chaque appel au fils droit reçoit `True`. En suivant l'arbre et ces valeurs, les affichages ont lieu dans l'ordre indiqué ci-dessous.

RÉPONSE À RETENIR

La fonction affiche, dans cet ordre, les prénoms suivants :

1. Odile
2. Hélène
3. Camélia
4. Marie
5. Eulalie
6. Gabrielle
7. Janet

FormaV - document d'entraînement non officiel

Exercice 5

4 points

Transmission de données dans un réseau, architecture d'un réseau, protocoles de routage, langages et programmation

1

Question 1.a

Sujet p. 13

> Combien d'octets sont nécessaires pour constituer une adresse IPv4 ?

Une adresse IPv4 comporte 32 bits, répartis en quatre groupes de 8 bits. Chaque groupe de 8 bits constitue un octet.

> **RÉPONSE À RETENIR**
>
> Il faut **4 octets** pour constituer une adresse IPv4.

Question 1.b

Sujet p. 13

> Quelle est la notation décimale du masque de sous-réseau du PC3 du réseau 1 ?

Le PC3 a une adresse en /24. Le masque comporte donc 24 bits à 1, puis 8 bits à 0 : 11111111.11111111.11111111.00000000. Chaque octet de 8 bits se convertit en décimal en 255, 255, 255 et 0.

> **RÉPONSE À RETENIR**
>
> La notation décimale du masque est **255.255.255.0**.

2

Question 2.a

Sujet p. 13, 14

> Compléter la ligne 2 du tableau : conversion de l'adresse IP décimale en adresse IP binaire sur 4 octets.

On convertit chacun des quatre octets en binaire sur 8 bits : 172 = 10101100, 150 = 10010110, 4 = 00000100 et 30 = 00011110. Les zéros initiaux sont nécessaires pour obtenir exactement 8 bits par octet.

> **RÉPONSE À RETENIR**
>
> Ligne 2 : **10101100.10010110.00000100.00011110**.

Question 2.b

Sujet p. 13, 14

> Compléter la ligne 3 du tableau : compléter le masque de sous-réseau en notation binaire.

Le masque /24 contient 24 bits à 1 suivis de 8 bits à 0. En les regroupant par octets, on obtient trois octets à 1 puis un octet à 0.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Ligne 3 : **11111111.11111111.11111111.00000000**.

Question 2.c

Sujet p. 13, 14

> Compléter la ligne 4 du tableau : compléter l'adresse du réseau en notation binaire à la suite d'un ET logique entre chaque bit de la ligne 2 et la ligne 3.

On effectue un ET logique entre l'adresse et le masque, bit par bit. Les trois premiers octets sont conservés car ils sont combinés avec 11111111. Le dernier octet devient 00000000 car il est combiné avec 00000000.

RÉPONSE À RETENIR

Ligne 4 : **10101100.10010110.00000100.00000000**.

Question 2.d

Sujet p. 13, 14

> Compléter la ligne 5 du tableau : compléter l'adresse IP décimale du réseau 1.

On convertit en décimal les octets binaires de la ligne 4 : 10101100 = 172, 10010110 = 150, 00000100 = 4 et 00000000 = 0.

RÉPONSE À RETENIR

Ligne 5 : l'adresse du réseau 1 est **172.150.4.0**.

3

Question 3.a — proposition 1

Sujet p. 13, 14

> Parmi les propositions permettant d'associer un 4ème PC client au réseau 1, déterminer en justifiant si l'adresse 172.154.4.30 pourrait être utilisée.

Avec le masque /24, le réseau 1 est 172.150.4.0/24 : les trois premiers octets de toute adresse de ce réseau doivent être 172.150.4. L'adresse proposée contient 154 comme deuxième octet ; elle n'appartient donc pas à ce réseau.

RÉPONSE À RETENIR

Non. 172.154.4.30 n'appartient pas au réseau 172.150.4.0/24.

Question 3.a — proposition 2

Sujet p. 13, 14

> Parmi les propositions permettant d'associer un 4ème PC client au réseau 1, déterminer en justifiant si l'adresse 172.150.4.10 pourrait être utilisée.

L'adresse 172.150.4.10 est bien dans le réseau 172.150.4.0/24 et correspond à une adresse d'hôte. Cependant, elle est déjà attribuée à une machine du réseau 1 dans le schéma de la page 13. La réutiliser créerait un doublon d'adresse.

RÉPONSE À RETENIR

Non. 172.150.4.10 est déjà utilisée par une machine du réseau 1.

Question 3.a — proposition 3

Sujet p. 13, 14

> Parmi les propositions permettant d'associer un 4ème PC client au réseau 1, déterminer en justifiant si l'adresse 172.150.10.257 pourrait être utilisée.

Chaque octet d'une adresse IPv4 doit être compris entre 0 et 255 inclus. Dans 172.150.10.257, le dernier octet vaut 257 et dépasse cette limite.

RÉPONSE À RETENIR

Non. 172.150.10.257 n'est pas une adresse IPv4 valide.

Question 3.a — proposition 4

Sujet p. 13, 14

> Parmi les propositions permettant d'associer un 4ème PC client au réseau 1, déterminer en justifiant si l'adresse 172.150.4.11 pourrait être utilisée.

Les trois premiers octets de 172.150.4.11 correspondent au réseau 172.150.4.0/24. Le dernier octet, 11, est compris entre 1 et 254 : ce n'est ni l'adresse du réseau ni l'adresse de diffusion. D'après le schéma, cette adresse n'est pas déjà attribuée.

RÉPONSE À RETENIR

Oui. 172.150.4.11 peut être utilisée pour le quatrième PC client.

Question 3.a — proposition 5

Sujet p. 13, 14

> Parmi les propositions permettant d'associer un 4ème PC client au réseau 1, déterminer en justifiant si l'adresse 172.150.4.0 pourrait être utilisée.

Avec le masque /24, 172.150.4.0 désigne l'adresse du réseau lui-même. Elle ne peut donc pas être attribuée à un PC.

RÉPONSE À RETENIR

Non. 172.150.4.0 est l'adresse du réseau.

Question 3.a — proposition 6

Sujet p. 13, 14

> Parmi les propositions permettant d'associer un 4ème PC client au réseau 1, déterminer en justifiant si l'adresse 172.150.4.200 pourrait être utilisée.

Les trois premiers octets de 172.150.4.200 correspondent au réseau 172.150.4.0/24. Le dernier octet, 200, est compris entre 1 et 254 et l'adresse n'est pas attribuée aux machines du schéma.

RÉPONSE À RETENIR

Oui. 172.150.4.200 peut être utilisée pour le quatrième PC client.

Question 3.b

Sujet p. 14

> Quelle commande permettrait de connaître son adresse IP ?

La commande dépend du système d'exploitation. Sous Windows, la commande usuelle est `ipconfig`. Sous Linux, on peut notamment utiliser `ip addr`.

> **RÉPONSE À RETENIR**
>
> Sous Windows, on peut saisir **`ipconfig`**.

Question 4.

Sujet p. 13, 14, 15

> On décide de connecter directement le switch 1 avec le switch 2 pour réaliser cette nouvelle configuration du réseau. Expliquer pourquoi cette solution n'est pas satisfaisante et proposer une alternative.

Un switch assure la communication au sein d'un réseau local, mais ne réalise pas le routage entre deux réseaux IP distincts. Relier directement les deux switches ne suffit donc pas pour que toutes les machines des deux réseaux puissent communiquer.

On peut relier chacun des switches à un routeur configuré pour router les paquets entre les deux réseaux. Il faut également que les machines disposent d'une passerelle par défaut adaptée. Un commutateur de niveau 3 configuré pour effectuer le routage constitue une autre solution.

> **RÉPONSE À RETENIR**
>
> La liaison directe des switches ne réalise pas le routage entre les deux réseaux IP. Il faut notamment les relier au moyen d'un **routeur configuré pour assurer la communication entre les réseaux**.

Question 5.

Sujet p. 15

> Écrire en langage Python la fonction adresse. Elle prend en paramètres l'adresse IP (sous la forme d'une liste) que l'on souhaite tester et une liste de listes contenant les adresses IP des machines du réseau. Si l'adresse testée ne figure pas dans la liste, la fonction l'ajoute à la liste et affiche « pas trouvée, ajoutée » ; si l'adresse testée figure dans la liste, elle affiche « trouvée ».

On teste si la liste `ip` est déjà un élément de `liste_IP`. Si elle y figure, on affiche le message demandé sans modifier la liste ; sinon, on l'ajoute avec `append`.

```
def adresse(ip, liste_IP):
    if ip in liste_IP:
        print("trouvée")
    else:
        liste_IP.append(ip)
        print("pas trouvée, ajoutée")
```

Par exemple, avec la liste fournie, l'appel `adresse([192, 168, 10, 3], liste_IP)` ajoute cette adresse et affiche « pas trouvée, ajoutée ». L'appel suivant avec `[192, 168, 10, 25]` affiche « trouvée » et laisse la liste inchangée.

> **RÉPONSE À RETENIR**
>
> La fonction ajoute une adresse absente de la liste et affiche le message correspondant ; si l'adresse est déjà présente, elle affiche « trouvée » sans modifier la liste.

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.