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 2023 • Centres étrangers • Jour 2

Code sujet **23-NSIJ2G11** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Adressage IP et protocoles de routage
9 réponses • sujet p. 2–4

Exercice 2 — Bases de données relationnelles et langage SQL
8 réponses • sujet p. 5–6

Exercice 3 — Programmation en Python, manipulation des chaînes de caractères, arbres binaires de recherche et parcours de liste
10 réponses • sujet p. 7–9

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

Adressage IP et protocoles de routage

Partie A : L'adressage IP

Question 1.a

Sujet p. 2

> Pour le réseau F, dont le masque est 255.255.255.0, une machine connectée au switch R5 possède l'adresse IPv4 192.168.5.3. Proposer une adresse IP valide pour le routeur F.

Le masque `255.255.255.0` signifie que les trois premiers octets identifient le réseau. L'adresse `192.168.5.3` appartient donc au réseau `192.168.5.0`. Le routeur doit avoir une adresse de ce réseau, distincte de l'adresse de la machine et des adresses réservées au réseau et à la diffusion.

> **RÉPONSE À RETENIR**
>
> Par exemple, le routeur F peut avoir l'adresse **`192.168.5.1`**, qui est valide sur le réseau `192.168.5.0/24`.

Question 1.b

Sujet p. 2

> Pour le réseau F, dont le masque est 255.255.255.0, indiquer le nombre maximum de machines que l'on pourra connecter sur ce réseau.

Le masque `255.255.255.0` laisse 8 bits pour la partie hôte. Il existe donc $2^8 = 256$ adresses dans le sous-réseau. Dans un sous-réseau IPv4 usuel, l'adresse dont la partie hôte vaut 0 désigne le réseau et celle dont la partie hôte vaut 255 est l'adresse de diffusion ; ces deux adresses ne sont pas attribuables aux machines.

> **RÉPONSE À RETENIR**
>
> Le nombre maximal de machines est **256 − 2 = 254**.

Question 2.a

Sujet p. 2, 3

> Pour le réseau B, identifier son masque de sous-réseau sur la figure 1.

On lit le masque inscrit dans l'encadré du réseau B sur la figure 1.

> **RÉPONSE À RETENIR**
>
> Le masque de sous-réseau du réseau B est **`255.255.240.0`**.

Question 2.b

Sujet p. 2, 3

> Déterminer l'adresse du réseau B à partir de l'adresse IP d'une machine de ce réseau, par exemple 192.168.2.2, et du masque de ce réseau, 255.255.240.0. Détailler soigneusement chaque étape du raisonnement.

On calcule l'adresse réseau par un ET logique entre chaque octet de l'adresse IP de la machine et l'octet correspondant du masque. Avec l'adresse `192.168.2.2` et le masque `255.255.240.0` :

- `192 ET 255 = 192` ;
- `168 ET 255 = 168` ;

FormaV - document d'entraînement non officiel

- pour le troisième octet, `2 = 00000010` et `240 = 11110000`, donc `00000010 ET 11110000 = 00000000`, soit 0 ;
- `2 ET 0 = 0` pour le quatrième octet.

> **RÉPONSE À RETENIR**
>
> L'adresse du réseau B est donc **`192.168.0.0`**, soit `192.168.0.0/20`.

Question 2.c

Sujet p. 2, 3

> Proposer un intérêt au fait d'avoir une telle interconnexion entre les quatre routeurs A, B, E et F.

L'interconnexion permet aux machines des différents réseaux de communiquer en faisant transiter les données par les routeurs. Comme plusieurs liaisons relient les routeurs, certains échanges peuvent aussi emprunter un autre chemin si une liaison ou un routeur devient indisponible.

> **RÉPONSE À RETENIR**
>
> Elle permet donc la communication entre les réseaux et peut offrir des chemins de remplacement en cas de panne.

Partie B : Le routage

Question 1.a

Sujet p. 3

> Dans le cadre du protocole RIP, qui emprunte le chemin ayant la plus petite distance en nombre de sauts, donner, en respectant ce protocole, le ou les chemins possibles entre les routeurs A et E, puis entre les routeurs F et B.

Avec RIP, on retient les chemins qui comportent le moins de liaisons, sans tenir compte de leur débit. De A à E, le chemin A-B-E compte 2 sauts. De F à B, les chemins minimaux comptent 3 sauts ; il faut donc donner tous les chemins de cette longueur visibles sur la figure.

> **RÉPONSE À RETENIR**
>
> - **De A à E :** A-B-E, en 2 sauts.
> - **De F à B :** F-D-A-B, F-D-G-B, F-H-G-B ou F-H-E-B ; chacun compte 3 sauts.

Question 1.b

Sujet p. 3, 4

> Recopier et compléter les tableaux de routage du routeur E et du routeur G, en indiquant les destinations, les routeurs suivants et les distances.

La distance RIP est le nombre de sauts. Pour chaque destination, le routeur suivant est le premier routeur rencontré sur un chemin de distance minimale. Lorsque plusieurs chemins minimaux existent, plusieurs routeurs suivants sont possibles.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Une complétion possible des tables est :

Table de routage du routeur E		
Destination	**Routeur suivant**	**Distance**
A	B	2
B	B	1
C	H	2
D	G	2
F	H	2
G	G	1
H	H	1

Table de routage du routeur G		
Destination	**Routeur suivant**	**Distance**
A	B ou D	2
B	B	1
C	D ou H	2
D	D	1
E	E	1
F	D ou H	2
H	H	1

Par exemple, E atteint C par E-H-C en 2 sauts ; depuis G, les chemins G-D-C et G-H-C comptent chacun 2 sauts.

Question 2.a

Sujet p. 3, 4

> Avec le protocole OSPF, qui se base sur le coût total minimal de la communication et calcule le coût entre deux routeurs selon la formule coût = 10^8 / débit, recopier et compléter la table de routage du routeur F. La table indique déjà : pour A, routeur suivant D et coût total 1,1 ; pour B, coût total 10,11 ; pour C, routeur suivant D et coût total 1,1 ; pour D, routeur suivant D ; pour E, routeur suivant H et coût total 10,1 ; pour G, routeur suivant D ; pour H, routeur suivant H et coût total 0,1.

Pour chaque liaison, on applique `coût = 10⁸ / débit`, avec le débit exprimé en bits par seconde. Ainsi, les liaisons de 10 Mbit/s coûtent 10, celles de 10 Gbit/s coûtent 0,01, celles de 100 Mbit/s coûtent 1 et celles de 1 Gbit/s coûtent 0,1. Le coût d'un chemin est la somme des coûts de ses liaisons. Dans la table, la destination D est le routeur lui-même : le coût pour l'atteindre est donc 0.

RÉPONSE À RETENIR

Quelques calculs permettent de compléter les coûts : vers B, le chemin F-H-E-B coûte `0,1 + 10 + 0,01 = 10,11` ; vers D, la liaison directe F-D coûte `0,1` ; vers G, le chemin F-D-G coûte `0,1 + 1 = 1,1`. La table complétée est :

Destination	Routeur suivant	Coût total
A	D	1,1
B	H	10,11
C	D	1,1
D	D	0
E	H	10,1
G	D	1,1
H	H	0,1

Pour G, le chemin F-H-G a lui aussi un coût de `0,1 + 1 = 1,1` ; le routeur suivant D indiqué dans la table est également un choix minimal.

Question 2.b

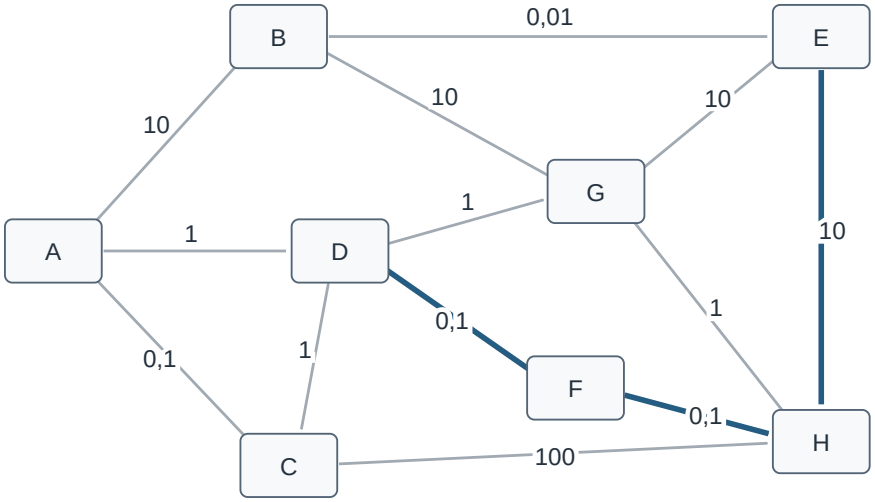
Sujet p. 3, 4

> Avec le protocole OSPF, indiquer quel sera le chemin emprunté par les informations entre le routeur E et le routeur D.

On compare les coûts totaux des chemins reliant E à D. Par E-H-F-D, le coût est `10 + 0,1 + 0,1 = 10,2`. Par E-G-D, il est `10 + 1 = 11`, et par E-B-A-D, il est `0,01 + 10 + 1 = 11,01`. Le chemin E-H-F-D est donc de coût minimal.

FormaV - document d'entraînement non officiel

Réseau de routage pondéré par les coûts OSPF



Les étiquettes indiquent les coûts OSPF calculés à partir des débits de la figure 2. Le chemin minimal E-H-F-D est mis en évidence.

RÉPONSE À RETENIR

Les informations empruntent le chemin **E-H-F-D**, de coût total **10,2**.

Exercice 2

4 points

Bases de données relationnelles et langage SQL

Question 1.a

Sujet p. 5, 6

> Écrire le résultat de la requête SQL : SELECT age, taille, poids FROM animal WHERE nom = 'Moris';

La requête sélectionne les colonnes `age`, `taille` et `poids` de la ligne dont le nom est Moris. Dans l'extrait de la table `animal`, cette ligne indique 6 ans, 1.70 m et 100 kg.

RÉPONSE À RETENIR

age	taille	poids
6	1.70	100

Question 1.b

Sujet p. 5, 6

> Écrire une requête qui permet d'obtenir le nom et l'âge de tous les animaux de l'espèce bonobo, triés du plus jeune au plus vieux.

On sélectionne le nom et l'âge dans la table `animal`, en ne conservant que les animaux dont `nom_espece` vaut `bonobo`. Le tri croissant sur `age` les classe du plus jeune au plus vieux.

```
SELECT nom, age
FROM animal
WHERE nom_espece = 'bonobo'
ORDER BY age;
```

RÉPONSE À RETENIR

Parmi les lignes affichées, Boris est le seul animal bonobo : il a 30 ans. L'extrait étant incomplet, il ne permet pas de connaître d'éventuelles autres lignes de la table.

nom	age
Boris	30

Question 2.a

Sujet p. 5, 6

> Citer, en justifiant, la clé primaire et la clé étrangère de la relation espece.

La clé primaire de `espece` est `nom_espece` : chaque ligne décrit une espèce, identifiée par son nom. La clé étrangère est `num_enclos` : elle désigne l'enclos associé à cette espèce et référence `enclos(num_enclos)`.

RÉPONSE À RETENIR

Clé primaire : **nom_espece**. Clé étrangère : `num_enclos`, qui référence la clé primaire `enclos(num_enclos)`.

FormaV - document d'entraînement non officiel

Question 2.b

Sujet p. 5, 6

> Donner le modèle relationnel de la base de données du zoo. Souligner les clés primaires et faire précéder les clés étrangères d'un #.

Les attributs de chaque relation sont repris des tables de l'énoncé. Dans le modèle ci-dessous, les clés primaires sont soulignées et le préfixe `#` signale une clé étrangère. L'attribut `nom_espece` de `animal` référence l'espèce de l'animal ; `num_enclos` de `espece` référence son enclos.

> **RÉPONSE À RETENIR**
>
> `animal(id_animal, nom, age, taille, poids, #nom_espece)`
>
> `enclos(num_enclos, ecosysteme, surface, struct, date_entretien)`
>
> `espece(nom_espece, classe, alimentation, #num_enclos)`

Question 3.a

Sujet p. 5, 6

> L'espèce ornithorynque a été entrée dans la base comme étant de la classe des oiseaux alors qu'il s'agit d'un mammifère. Écrire une requête qui corrige cette erreur dans la table espece.

Il faut mettre à jour l'attribut `classe` de la ligne correspondant à l'espèce ornithorynque. La condition sur `nom_espece` limite la modification à cette espèce.

```
UPDATE espece
SET classe = 'mammifères'
WHERE nom_espece = 'ornithorynque';
```

> **RÉPONSE À RETENIR**
>
> La classe de l'espèce ornithorynque est alors enregistrée comme `mammifères`.

Question 3.b

Sujet p. 5, 6

> Le couple de lamas du zoo vient de donner naissance au petit lama nommé « Serge », qui mesure 80 cm et pèse 30 kg. Écrire une requête qui permet d'enregistrer ce nouveau venu au zoo dans la base de données, sachant que les clés primaires de 1 à 178 sont déjà utilisées.

Les identifiants de 1 à 178 étant déjà utilisés, on choisit 179, en supposant qu'il est libre. La taille est enregistrée en mètres : 80 cm = 0.80 m. L'âge du nouveau-né n'étant pas précisé, on prend la convention qu'il vaut 0 an. La liste des colonnes rend explicite l'ordre des valeurs insérées.

```
INSERT INTO animal (id_animal, nom, age, taille, poids, nom_espece)
VALUES (179, 'Serge', 0, 0.80, 30, 'lama');
```

> **RÉPONSE À RETENIR**
>
> La ligne insérée enregistre Serge sous l'identifiant 179, avec un âge de 0 an, une taille de 0.80 m, un poids de 30 kg et l'espèce `lama`.

FormaV - document d'entraînement non officiel

Question 4.a

Sujet p. 5, 6

> Recopier la requête SQL et compléter les emplacements manquants afin de recenser le nom et l'espèce de tous les animaux carnivores vivant en vivarium dans le zoo : SELECT ... FROM animal JOIN espece ON ... JOIN enclos ON ... WHERE enclos.struct = 'vivarium' and ...;

La première jointure relie l'animal à son espèce par `nom_espece`. La seconde relie l'espèce à son enclos par `num_enclos`. Les deux filtres imposent une alimentation carnivore et une structure d'enclos de type vivarium.

```
SELECT animal.nom, animal.nom_espece
FROM animal
JOIN espece ON animal.nom_espece = espece.nom_espece
JOIN enclos ON espece.num_enclos = enclos.num_enclos
WHERE enclos.struct = 'vivarium' AND espece.alimentation = 'carnivore';
```

RÉPONSE À RETENIR

Sur les seuls enregistrements affichés, aucune ligne d'animal ne satisfait simultanément les jointures et les conditions. Les extraits étant incomplets, cela ne permet pas de conclure qu'il n'existe aucun animal correspondant dans le zoo entier.

nom	nom_espece

Question 4.b

Sujet p. 5, 6

> Écrire la requête qui permet de compter le nombre d'animaux dans tout le zoo qui font partie de la classe des oiseaux.

Il faut relier chaque animal à son espèce, puis ne conserver que les espèces de classe `oiseaux`. `COUNT(*)` compte alors les animaux correspondants, et non les espèces elles-mêmes.

```
SELECT COUNT(*)
FROM animal
JOIN espece ON animal.nom_espece = espece.nom_espece
WHERE espece.classe = 'oiseaux';
```

RÉPONSE À RETENIR

En se limitant aux lignes affichées et aux correspondances disponibles entre les extraits, le compte est nul. Comme les extraits sont incomplets, le nombre total d'animaux de classe oiseaux dans le zoo ne peut pas être déterminé à partir de ces seules données.

COUNT(*) sur les extraits
0

FormaV - document d'entraînement non officiel

Exercice 3

4 points

Programmation en Python, manipulation des chaînes de caractères, arbres binaires de recherche et parcours de liste

Question 1.a

Sujet p. 7

> La fonction bonjour est définie par le code `def bonjour(nom)` puis `return 'Bonjour ' + nom + ' !'`. Donner le résultat de l'exécution de bonjour('Alan').

La fonction concatène la chaîne `'Bonjour '`, le nom fourni et la chaîne `' !'`. Les espaces font partie des chaînes concaténées.

RÉPONSE À RETENIR

`bonjour('Alan')` renvoie la chaîne `'Bonjour Alan !'`.

Question 1.b

Sujet p. 7

> On exécute le programme `une_chaine = 'Bonjour'`, `x = (une_chaine[2] == une_chaine[3])` et `y = (une_chaine[4] == une_chaine[1])`. Donner le type et les valeurs des variables x et y.

Les indices d'une chaîne commencent à 0. Dans `'Bonjour'`, on a `une_chaine[2] == 'n'`, `une_chaine[3] == 'j'`, `une_chaine[4] == 'o'` et `une_chaine[1] == 'o'`. Les comparaisons produisent des valeurs booléennes.

RÉPONSE À RETENIR

`x` est de type `bool` et vaut `False`, car `'n' != 'j'`. `y` est de type `bool` et vaut `True`, car `'o' == 'o'`.

Question 1.c

Sujet p. 7

> Écrire une fonction occurrences_lettre(une_chaine, une_lettre) prenant en paramètres une chaîne une_chaine et une lettre une_lettre et renvoyant le nombre d'occurrences de une_lettre dans une_chaine.

On parcourt chaque caractère de la chaîne et on augmente un compteur chaque fois qu'il est égal à la lettre recherchée. Le compteur est initialisé à 0 : si la chaîne est vide, la fonction renvoie donc bien 0.

```
def occurrences_lettre(une_chaine, une_lettre):
    nombre = 0
    for caractere in une_chaine:
        if caractere == une_lettre:
            nombre += 1
    return nombre
```

La boucle examine chaque caractère une seule fois : la fonction termine et son temps d'exécution est proportionnel à la longueur de la chaîne.

RÉPONSE À RETENIR

La fonction ci-dessus renvoie le nombre d'occurrences de `une_lettre` dans `une_chaine`.

FormaV - document d'entraînement non officiel

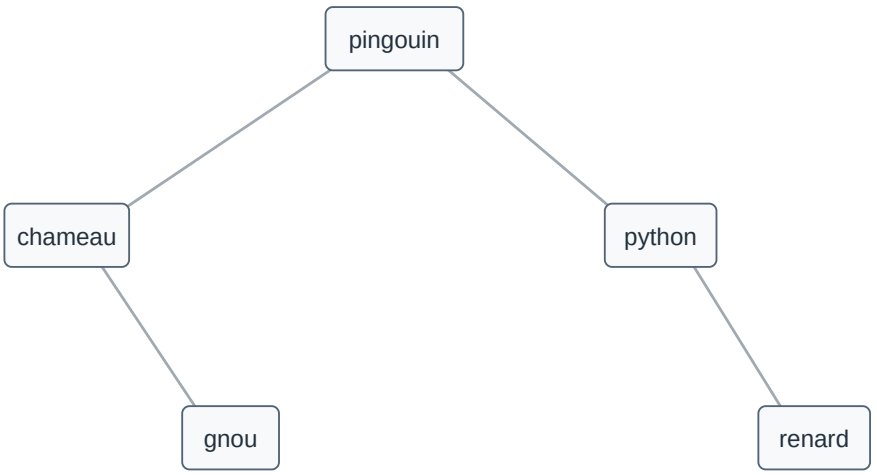
Question 2.a

Sujet p. 8

> On considère la liste de mots animaux = ['python', 'chameau', 'pingouin', 'renard', 'gnou']. Dessiner un arbre binaire de recherche contenant tous les mots de cette liste et de hauteur minimale, en utilisant l'ordre alphabétique pour comparer les chaînes.

Dans l'ordre alphabétique, les mots sont rangés ainsi : `chameau`, `gnou`, `pingouin`, `python`, `renard`. Avec 5 nœuds, la hauteur minimale est de 3 niveaux : un arbre binaire de hauteur 2 ne peut contenir que 3 nœuds au maximum. On choisit `pingouin` comme racine. À gauche, `chameau` a pour fils droit `gnou` ; à droite, `python` a pour fils droit `renard`. Chaque mot est ainsi placé conformément à l'ordre d'un arbre binaire de recherche.

Arbre binaire de recherche de hauteur minimale



Les fils gauche et droit respectent l'ordre alphabétique.

> **RÉPONSE À RETENIR**
>
> Un arbre binaire de recherche de hauteur minimale est représenté ci-dessous. Sa hauteur est de 3 niveaux.

Question 2.b

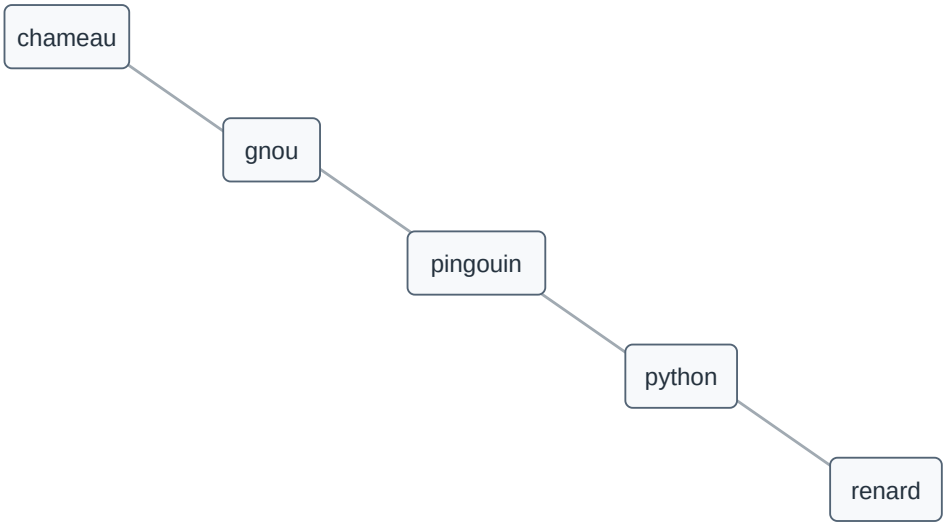
Sujet p. 8

> On considère la liste de mots animaux = ['python', 'chameau', 'pingouin', 'renard', 'gnou']. Dessiner un arbre binaire de recherche contenant tous ces mots et de hauteur maximale, en utilisant l'ordre alphabétique pour comparer les chaînes.

Pour obtenir la hauteur maximale, on place les mots dans une chaîne de fils droits, dans leur ordre alphabétique. Chaque nouveau mot est alors supérieur à celui du nœud précédent et se place bien dans son sous-arbre droit. Les cinq nœuds occupent cinq niveaux.

FormaV - document d'entraînement non officiel

Arbre binaire de recherche de hauteur maximale



Chaque mot est le fils droit du précédent, conformément à l'ordre alphabétique.

> **RÉPONSE À RETENIR**
>
> Un arbre binaire de recherche de hauteur maximale a une hauteur de 5 niveaux.

Question 3.a

Sujet p. 8, 9

> La liste liste_mots_francais contient 336531 mots et abr_mots_francais est une instance d'Abr contenant les mots de cette liste. La fonction mystere renvoie 0 si l'arbre est vide et, sinon, 1 plus le résultat de son application à chacun de ses deux sous-arbres. Donner le résultat de mystere(abr_mots_francais), en justifiant le résultat.

Pour un arbre vide, `mystere` renvoie 0. Pour un arbre non vide, la fonction ajoute 1 pour sa racine, puis compte les nœuds de chacun des deux sous-arbres. Ainsi, chaque nœud de l'arbre contribue exactement une fois à la somme. Comme `abr_mots_francais` contient les 336531 mots, le résultat mathématique est 336531.

La fonction récursive donnée parcourt tous les nœuds, donc son temps d'exécution est proportionnel au nombre de nœuds. Sa profondeur d'appels dépend de la hauteur de l'arbre : sans connaître cette hauteur, on ne peut pas garantir qu'un appel Python sur l'arbre fourni évitera une limite de récursion. Un parcours itératif permet d'éviter cette dépendance à la pile d'appels.

```
def mystere_iteratif(un_abr):
    nombre = 0
    pile = [un_abr]
    while pile:
        courant = pile.pop()
        if not courant.est_vide():
            nombre += 1
            pile.append(courant.sous_arbre_gauche)
            pile.append(courant.sous_arbre_droit)
    return nombre
```

RÉPONSE À RETENIR

`mystere(abr_mots_francais)` compte les nœuds de l'arbre et renvoie donc **336531**, si l'appel récursif s'exécute sans dépasser la limite de récursion de Python.

Question 3.b

Sujet p. 8, 9

> On veut calculer la hauteur de abr_mots_francais, la hauteur étant le nombre de niveaux de l'arbre. Donner le code d'une fonction hauteur(un_abr) permettant de faire ce calcul, en s'inspirant du code de mystere.

La hauteur est le nombre de niveaux. Un arbre vide a une hauteur de 0. Pour un arbre non vide, sa racine constitue un niveau et la hauteur de chacun de ses sous-arbres est calculée récursivement ; on ajoute 1 à la plus grande des deux hauteurs. Les appels se terminent car ils portent sur des sous-arbres de plus en plus petits jusqu'à atteindre des arbres vides.

```
def hauteur(un_abr):
    if un_abr.est_vide():
        return 0
    else:
        return 1 + max(
            hauteur(un_abr.sous_arbre_gauche),
            hauteur(un_abr.sous_arbre_droit)
        )
```

Cette fonction parcourt tous les nœuds, en temps proportionnel à leur nombre, mais sa profondeur d'appels peut atteindre la hauteur de l'arbre. Si celle-ci risque de dépasser la limite de récursion de Python, on peut calculer la même hauteur par un parcours itératif.

```
def hauteur_iterative(un_abr):
    if un_abr.est_vide():
        return 0
    pile = [(un_abr, 1)]
    maximum = 0
    while pile:
        courant, niveau = pile.pop()
        maximum = max(maximum, niveau)
        gauche = courant.sous_arbre_gauche
        droit = courant.sous_arbre_droit
        if not gauche.est_vide():
            pile.append((gauche, niveau + 1))
        if not droit.est_vide():
            pile.append((droit, niveau + 1))
    return maximum
```

Dans cette version, chaque nœud non vide est traité une fois ; la pile explicite évite d'accumuler les appels récursifs.

RÉPONSE À RETENIR

La fonction `hauteur` ci-dessus renvoie 0 pour un arbre vide et, sinon, 1 plus la hauteur du plus profond de ses deux sous-arbres.

Question 4.a

Sujet p. 9

> Dans cette question, utiliser uniquement liste_mots_francais et non abr_mots_francais. Recopier et compléter la ligne 4 de la fonction chercher_mots(liste_mots, longueur, lettre, position), qui recherche les mots de la longueur indiquée ayant la lettre donnée à l'indice position : `if … and … :`.

Il faut vérifier les deux critères : le mot a la longueur demandée, et le caractère situé à l'indice `position` est égal à `lettre`. En Python, les indices des caractères commencent à 0. La condition de longueur est testée en premier ; avec l'évaluation court-circuitée de `and`, un mot de longueur différente n'est pas indexé.

RÉPONSE À RETENIR

```
if len(liste_mots[i]) == longueur and liste_mots[i][position] == lettre:
```

Question 4.b

Sujet p. 9

> Expliquer ce que donne la commande `chercher_mots(chercher_mots(liste_mots_francais, 3, 'x', 2), 3, 'a', 1)`.

L'appel intérieur sélectionne les mots de 3 lettres dont le caractère d'indice 2, donc la dernière lettre, est `'x'`. L'appel extérieur conserve parmi ces mots ceux dont le caractère d'indice 1, donc la lettre du milieu, est `'a'`. Dans la liste donnée dans l'énoncé, ces deux critères sont satisfaits par `'fax'` et `'max'`, dans cet ordre.

RÉPONSE À RETENIR

La commande renvoie `['fax', 'max']`.

Question 4.c

Sujet p. 9

> On cherche un mot de 5 lettres dont on connaît la fin « ter ». Écrire la commande permettant de chercher les mots candidats dans liste_mots_francais.

Dans un mot de 5 lettres finissant par `'ter'`, les lettres recherchées sont `'t'` à l'indice 2, `'e'` à l'indice 3 et `'r'` à l'indice 4. Comme `chercher_mots` ne teste qu'une position à la fois, on enchaîne trois appels en filtrant successivement la liste obtenue.

RÉPONSE À RETENIR

```
chercher_mots(
    chercher_mots(
        chercher_mots(liste_mots_francais, 5, 't', 2),
        5, 'e', 3
    ),
    5, 'r', 4
)
```

Sujet : https://www.sujetdebac.fr/annales-pdf/2023/spe-numerique-informatique-2023-centres-etranger-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.