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 2

Code sujet **22-NSIJ2LR1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Structures de données : pile et notation polonaise inverse
5 réponses • sujet p. 2–3

Exercice 2 — Bases de données et requêtes SQL
6 réponses • sujet p. 4–12

Exercice 3 — Représentation binaire d’un entier relatif et systèmes d’exploitation
9 réponses • sujet p. 6–13

Exercice 4 — Arbres binaires de recherche
5 réponses • sujet p. 8–9

Exercice 5 — Algorithmes et programmation Python : jeu du TAKAZU
6 réponses • sujet p. 10–11

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 : pile et notation polonaise inverse

Question 1.

Sujet p. 2

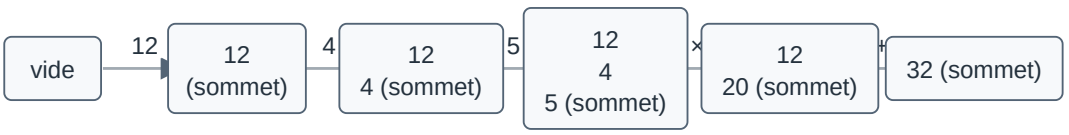
> En vous inspirant de l'exemple, dessiner le schéma descriptif de l'évaluation par la notation polonaise inverse de l'expression 12 4 5 × +.

On lit l'expression de gauche à droite. Chaque nombre est empilé ; à chaque opérateur, on dépile les deux valeurs au sommet, on effectue l'opération, puis on empile le résultat. Les piles ci-dessous sont présentées du fond vers le sommet.

Élément lu	État de la pile
Départ	vide
12	12
4	12, 4
5	12, 4, 5
×	12, 20
+	32

À la lecture de ×, les deux valeurs du sommet sont 5 et 4 : leur produit est 20. À la lecture de +, on additionne 20 et 12.

États successifs de la pile pour 12 4 5 × +



Les piles sont indiquées du fond vers le sommet.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

L'expression **12 4 5 × +** donne **32**. La pile finale contient uniquement 32.

a.

Question 2.a.

Sujet p. 3

On dispose de la pile p1 représentée sur le sujet, contenant 7, 3 et 25, avec 25 au sommet. Après l'exécution de temp = top(p1), quelle valeur contient la variable temp ?

La fonction `top(p)` dépile l'élément au sommet, le réempile aussitôt, puis le renvoie. Le sommet de `p1` est 25 : cette valeur est donc affectée à `temp`.

RÉPONSE À RETENIR

`temp` contient **25**.

b.

Question 2.b.

Sujet p. 3

Représenter la pile p1 après l'exécution de temp = top(p1).

La fonction `top(p1)` réempile l'élément qu'elle vient de dépiler. Elle ne modifie donc pas le contenu de la pile, même si elle a temporairement retiré son sommet. L'ordre reste le même : 7 au fond, puis 3, et 25 au sommet.

RÉPONSE À RETENIR

Après l'exécution, `p1` contient toujours, du fond vers le sommet, **7, 3, 25**.

Question 3.

Sujet p. 3

En utilisant uniquement les quatre primitives d'une pile, écrire en langage Python la fonction addition(p), qui prend en paramètre une pile p d'au moins deux éléments et remplace les deux nombres au sommet de p par leur somme. La fonction ne renvoie rien, mais la pile p est modifiée.

On dépile les deux nombres au sommet, on les additionne, puis on empile leur somme. Comme l'addition est commutative, l'ordre dans lequel ces deux nombres sont dépilés ne change pas le résultat. La fonction ne comporte pas de `return` : elle modifie directement la pile passée en paramètre.

RÉPONSE À RETENIR

```
def addition(p):
    x = depiler(p)
    y = depiler(p)
    empiler(p, x + y)
```

Les deux valeurs du sommet sont remplacées par leur somme ; les éléments situés en dessous ne sont pas modifiés.

Question 4.

Sujet p. 3

> On dispose d'une fonction multiplication(p) qui remplace les deux nombres au sommet de p par leur produit. Recopier et compléter la suite d'instructions en n'utilisant que les primitives d'une pile et les fonctions addition et multiplication, afin de réaliser le calcul (3 + 5) × 7, dont l'écriture en NPI est 3 5 + 7 ×. La suite fournie commence par p = pile_vide() puis empiler(p, 3).

Pour évaluer `3 5 + 7 ×`, on empile d'abord 3 puis 5. L'appel à `addition(p)` remplace ces deux valeurs par 8. On empile ensuite 7, puis `multiplication(p)` remplace 8 et 7 par leur produit.

En complétant la suite fournie, on obtient :

RÉPONSE À RETENIR

```
p = pile_vide()
empiler(p, 3)
empiler(p, 5)
addition(p)
empiler(p, 7)
multiplication(p)
```

Étape	Pile, du fond vers le sommet
Départ	vide
Empiler 3	3
Empiler 5	3, 5
`addition(p)`	8
Empiler 7	8, 7
`multiplication(p)`	56

La pile contient finalement 56, car **(3 + 5) × 7 = 56**.

FormaV - document d'entraînement non officiel

Exercice 2

4 points

Bases de données et requêtes SQL

Question 1.

Sujet p. 4

> Expliquer pourquoi le couple (NumClient, NumChambre) ne pouvait pas servir de clé primaire pour la relation Reservations.

Une clé primaire doit distinguer chaque ligne de la relation. Or le même client peut réserver la même chambre à plusieurs dates différentes : le couple (NumClient, NumChambre) serait alors identique pour plusieurs réservations. Le schéma utilise donc NumRes comme clé primaire pour identifier chaque réservation.

> **RÉPONSE À RETENIR**
>
> Le couple (NumClient, NumChambre) ne peut pas servir de clé primaire, car il peut se répéter pour des réservations différentes du même client dans la même chambre.

a.

Question 2.a.

Sujet p. 4, 12

> Écrire une requête SQL donnant la liste des noms et prénoms des clients.

Les attributs demandés, Nom et Prenom, se trouvent dans la relation Clients. Il suffit donc de sélectionner ces deux colonnes dans cette relation.

```
SELECT Nom, Prenom
FROM Clients;
```

> **RÉPONSE À RETENIR**
>
> La requête renvoie les noms et prénoms de tous les clients. Aucun extrait de données n'étant fourni, les lignes du résultat ne peuvent pas être précisées.

b.

Question 2.b.

Sujet p. 4, 12

> Écrire une requête SQL donnant le numéro de téléphone d'une cliente s'appelant « Grace Hopper ».

Dans le schéma, Nom désigne le nom de famille et Prenom le prénom. Pour Grace Hopper, on filtre donc sur Prenom = 'Grace' et Nom = 'Hopper', puis on affiche Telephone.

```
SELECT Telephone
FROM Clients
WHERE Prenom = 'Grace' AND Nom = 'Hopper';
```

RÉPONSE À RETENIR

La requête renvoie le numéro de téléphone enregistré pour Grace Hopper. Aucun extrait de données n'étant fourni, ce numéro ne peut pas être précisé.

Question 3.

Sujet p. 4, 12

> Les dates DateArr et DateDep sont au format chaîne de caractères « aaaa-mm-jj » et peuvent être manipulées avec la fonction date(). Un client nommé « JOHN DOE » veut réserver une chambre pour la nuit du 28 décembre 2024. Écrire une requête donnant la liste des numéros des chambres occupées à cette date.

Une réservation occupe la chambre pendant la nuit du 28 décembre si elle a commencé au plus tard le 28 décembre et si la chambre n'est libérée qu'après cette nuit. On adopte la convention usuelle de l'hôtel selon laquelle la date de départ est le jour de libération de la chambre : elle n'est donc pas une nuit occupée. Le nom JOHN DOE indique le contexte de la demande ; la question demande les numéros de toutes les chambres occupées à cette date, et non uniquement celles réservées par ce client.

```
SELECT NumChambre
FROM Reservations
WHERE date(DateArr) <= date('2024-12-28')
  AND date(DateDep) > date('2024-12-28');
```

RÉPONSE À RETENIR

La requête sélectionne les numéros des chambres occupées pendant la nuit du 28 décembre 2024, selon la convention précisée. Aucun extrait de données n'étant fourni, les numéros obtenus ne peuvent pas être précisés.

a.

Question 4.a.

Sujet p. 4, 5, 12

> Le prix de la chambre 404 a changé et s'élève désormais à 75 euros. Quelle requête faut-il faire pour modifier ce prix dans la base de données ?

Le prix est un attribut de la relation Chambres. On modifie la ligne correspondant au numéro de chambre 404 avec une clause WHERE afin de ne pas changer le prix des autres chambres.

```
UPDATE Chambres
SET Prix = 75
WHERE NumChambre = 404;
```

RÉPONSE À RETENIR

Cette requête fixe à 75 euros le prix de la chambre 404.

b.

Question 4.b.

Sujet p. 4, 5, 12

> Écrire une requête SQL affichant les numéros des chambres où a séjourné « Edgar Codd ».

Dans le schéma, Nom est le nom de famille et Prenom le prénom : pour Edgar Codd, on sélectionne donc Nom = 'Codd' et Prenom = 'Edgar'. Les informations du client se trouvent dans Clients et les numéros des chambres dans Reservations ; la jointure se fait sur NumClient.

```
SELECT Reservations.NumChambre
FROM Clients JOIN Reservations
  ON Clients.NumClient = Reservations.NumClient
WHERE Clients.Nom = 'Codd' AND Clients.Prenom = 'Edgar';
```

RÉPONSE À RETENIR

La requête affiche les numéros des chambres associées aux réservations d’Edgar Codd. Aucun extrait de données n’étant fourni, les numéros obtenus ne peuvent pas être précisés.

Exercice 3

4 points

Représentation binaire d'un entier relatif et systèmes d'exploitation

a.

Question 1.a.

Sujet p. 6

> Codage des entiers naturels : combien de bits sont utilisés pour coder un entier naturel sur un octet ?

Un octet est composé de 8 bits. Chaque bit peut prendre l'une des deux valeurs 0 ou 1.

> **RÉPONSE À RETENIR**
>
> Un entier naturel codé sur un octet utilise **8 bits**.

b.

Question 1.b.

Sujet p. 6

> En déduire le nombre de valeurs pouvant être codées sur un octet.

Chaque bit ayant deux valeurs possibles, 8 bits permettent 2^8 combinaisons.

> **RÉPONSE À RETENIR**
>
> On peut coder $2^8 = 256$ valeurs différentes sur un octet.

c.

Question 1.c.

Sujet p. 6

> Donner un encadrement de ces valeurs.

Pour un entier naturel codé sans signe, la plus petite valeur est obtenue avec huit bits à 0 et la plus grande avec huit bits à 1. Cette dernière vaut $2^8 - 1 = 255$.

> **RÉPONSE À RETENIR**
>
> Les valeurs codables sont les entiers de 0 à 255 inclus : $0 \leq n \leq 255$.

a.

Question 2.a.

Sujet p. 6

> Pour effectuer sur 8 bits la soustraction 65 − 58, un processeur effectue l'addition 65 + (−58). Donner l'écriture en base 2 sur 8 bits du nombre 65.

On décompose $65 = 64 + 1 = 2^6 + 2^0$. Les bits correspondant à 64 et à 1 sont donc à 1 ; on complète l'écriture à huit bits avec des zéros à gauche.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

$65_{10} = 01000001_2$ sur 8 bits.

b.

Question 2.b.

Sujet p. 6

> Vérifier que 58 s'écrit 0011 1010 en base 2.

On décompose $58 = 32 + 16 + 8 + 2$. En plaçant les bits correspondant à ces puissances de 2 à 1, on obtient bien l'écriture demandée.

RÉPONSE À RETENIR

$58_{10} = 00111010_2$, soit **0011 1010** en groupes de quatre bits.

c.

Question 2.c.

Sujet p. 6

> En suivant le protocole du complément à deux indiqué dans le sujet pour coder un entier négatif sur 8 bits — coder la valeur absolue en base 2, compléter éventuellement l'octet à gauche avec des 0, échanger tous les bits 0 en 1 et réciproquement, puis additionner 00000001 — déterminer l'écriture de −58 sur 8 bits.

On écrit d'abord 58 sur 8 bits : `00111010`. On inverse les bits, puis on ajoute `00000001` conformément au protocole :

```
  11000101
+ 00000001
----------
  11000110
```

RÉPONSE À RETENIR

L'écriture de -58 sur 8 bits en complément à deux est **11000110**.

d.

Question 2.d.

Sujet p. 6

> Effectuer en binaire la soustraction de 65 − 58 telle que la ferait le processeur.

On additionne l'écriture sur 8 bits de 65 et celle de -58. La retenue qui dépasse les 8 bits est écartée, car le calcul est effectué sur 8 bits.

```
  01000001   (65)
+ 11000110   (-58)
----------
1 00000111
```

On conserve les huit bits de droite : `00000111`, qui vaut 7 en base 10.

RÉPONSE À RETENIR

$65 - 58 = 00000111_2 = 7.$

a.

Question 3.a.

Sujet p. 6, 7, 13

> Un disque dur contient l'arborescence A et doit finalement contenir l'arborescence B. Sachant que le dossier en cours est le dossier home, quelles commandes Linux faut-il saisir pour déplacer le fichier « saxo.mp3 » du dossier « documents » vers le dossier « musiques » ?

Dans l'arborescence, le répertoire courant `home` contient le dossier `pierre`, qui contient `documents` et `musiques`. Le fichier est donc dans `pierre/documents` et sa destination est `pierre/musiques`. La commande `mv` reçoit d'abord le chemin source, puis le chemin de destination.

RÉPONSE À RETENIR

```
mv pierre/documents/saxo.mp3 pierre/musiques/
```

b.

Question 3.b.

Sujet p. 6, 7, 13

> Un disque dur contient l'arborescence A et doit finalement contenir l'arborescence B. Sachant que le dossier en cours est le dossier home, quelles commandes Linux faut-il saisir pour renommer le dossier « bizarre » en dossier « videos » ?

Depuis le répertoire courant `home`, le dossier `bizarre` se trouve dans `pierre`. La commande `mv` permet de le renommer en indiquant son chemin actuel, puis son nouveau chemin.

RÉPONSE À RETENIR

```
mv pierre/bizarre pierre/videos
```

Exercice 4

4 points

Arbres binaires de recherche

a.

Question 1.a.

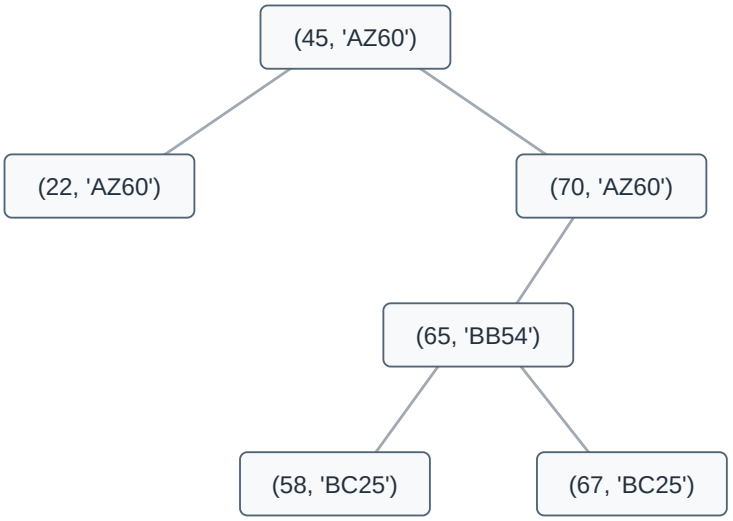
Sujet p. 8

> Dessiner l'arbre binaire de recherche dont la racine est (45, 'AZ60') et dans lequel on insère, dans cet ordre, les nœuds (70, 'AZ60'), (22, 'AZ60'), (65, 'BB54'), (58, 'BC25') et (67, 'BC25'). Seul le numéro du billet est utilisé pour positionner le nœud dans l'arbre.

On insère les nœuds dans l'ordre indiqué, en comparant à chaque étape le numéro du billet avec celui du nœud rencontré. Un numéro inférieur va à gauche ; un numéro supérieur ou égal va à droite.

1. 70 est supérieur à 45 : il devient le fils droit de 45.
2. 22 est inférieur à 45 : il devient le fils gauche de 45.
3. 65 est supérieur à 45, puis inférieur à 70 : il devient le fils gauche de 70.
4. 58 est supérieur à 45, inférieur à 70, puis inférieur à 65 : il devient le fils gauche de 65.
5. 67 est supérieur à 45, inférieur à 70, puis supérieur à 65 : il devient le fils droit de 65.

Arbre binaire de recherche obtenu après les insertions



Les fils gauches portent des numéros strictement inférieurs à leur parent ; les fils droits, des numéros supérieurs ou égaux.

RÉPONSE À RETENIR

L'arbre obtenu est représenté ci-dessous. Chaque nœud conserve aussi la référence du lieu de vente donnée dans l'énoncé.

FormaV - document d'entraînement non officiel

b.

Question 1.b.

Sujet p. 8

> Pour construire la liste triée de tous les numéros de billets vendus, préciser quel type de parcours de cet arbre il faut programmer parmi les propositions suivantes : un parcours en largeur, un parcours en profondeur infixe, un parcours en profondeur préfixe ou un parcours en profondeur suffixe.

Dans un arbre binaire de recherche, un parcours en profondeur infixe visite successivement le sous-arbre gauche, le nœud, puis le sous-arbre droit. Comme tous les numéros du sous-arbre gauche sont inférieurs au numéro du nœud et ceux du sous-arbre droit lui sont supérieurs ou égaux, ce parcours produit les numéros dans l'ordre croissant.

> **RÉPONSE À RETENIR**
>
> Il faut programmer un **parcours en profondeur infixe**. Pour l'arbre de la question 1.a, la liste obtenue est `[22, 45, 58, 65, 67, 70]`.

Question 2.

Sujet p. 9

> On rappelle que la taille d'un arbre est le nombre de nœuds. Recopier et compléter les blancs de l'algorithme de la fonction taille(a) pour que celle-ci renvoie la taille d'un arbre a. Les fonctions filsgauche(a) et filsdroit(a) renvoient les sous-arbres gauche et droit du nœud a, ou null si le nœud n'a pas le fils correspondant.

Un arbre vide contient zéro nœud. Pour un arbre non vide, on compte sa racine, puis on ajoute le nombre de nœuds de chacun de ses deux sous-arbres. Les appels récursifs s'appliquent à des sous-arbres de plus en plus petits et finissent donc par atteindre le cas vide.

> **RÉPONSE À RETENIR**
>
> ```
> fonction taille(a)
>     si a est null
>         alors renvoyer 0
>     sinon
>         renvoyer 1 + taille(filsgauche(a)) + taille(filsdroit(a))
> ```
>
> Par exemple, pour l'arbre de la question 1.a, la fonction renvoie 6 : sa racine compte pour 1 et ses sous-arbres contiennent au total 5 nœuds.

a.

Question 3.a.

Sujet p. 9

> Si a n'est pas null, billet(a) retourne le numéro du billet du nœud a. Que fait la fonction récursive mystere(a, n) écrite en pseudo-langage, où a est un nœud et n un nombre entier ?

Si le nœud est vide, la fonction renvoie Faux. Sinon, elle compare son numéro de billet à `n` : elle renvoie Vrai en cas d'égalité. En cas de différence, elle recherche récursivement dans les deux sous-arbres ; l'opérateur OU donne Vrai si l'un des deux appels trouve le numéro.

RÉPONSE À RETENIR

`mystere(a, n)` renvoie donc Vrai si et seulement si un nœud de l'arbre binaire contient le numéro de billet `n` ; sinon, elle renvoie Faux. Cette recherche examine potentiellement les nœuds des deux sous-arbres, car elle est conçue pour un arbre binaire quelconque.

b.

Question 3.b.

Sujet p. 8, 9

La fonction précédente est applicable à tout arbre binaire. L'arbre construit à la première question est un arbre binaire de recherche. Recopier et compléter le blanc de la ligne 6 de l'algorithme mystereABR(a, n) pour améliorer la fonction en tenant compte de cette remarque.

Dans un arbre binaire de recherche, si le numéro du nœud courant est supérieur à `n`, alors `n` ne peut se trouver que dans le sous-arbre gauche. Si le numéro courant est inférieur à `n`, il ne peut se trouver que dans le sous-arbre droit. Le cas d'égalité a déjà été traité à la ligne 4.

RÉPONSE À RETENIR

La condition à inscrire à la ligne 6 est `billet(a) > n`.

```
1. fonction mystereABR(a, n)
2.     si a est null
3.         alors renvoyer Faux
4.     sinon, si billet(a) vaut n
5.         alors renvoyer Vrai
6.     sinon si billet(a) > n
7.         renvoyer mystereABR(filsgauche(a))
8.     sinon
9.         renvoyer mystereABR(filsdroit(a))
```

Ainsi, à chaque étape, on écarte un des deux sous-arbres, ce qui évite la recherche exhaustive utilisée par `mystere`.

Exercice 5

4 points

Algorithmes et programmation Python : jeu du TAKAZU

Question 1.

Sujet p. 10

> Écrire une fonction autre(x) qui renvoie 1 si x vaut 0 et 0 si x vaut 1.

La fonction distingue les deux valeurs possibles : si `x` vaut 0, elle renvoie 1 ; sinon, pour une entrée valant 1, elle renvoie 0.

RÉPONSE À RETENIR

```
def autre(x):
    if x == 0:
        return 1
    return 0
```

Ainsi, `autre(0)` vaut 1 et `autre(1)` vaut 0.

a.

Question 2.a.

Sujet p. 10

> Écrire le code d'une fonction nbValeurs(li, v), dont les paramètres sont li, le numéro de la ligne de la grille, et v, la valeur que l'on souhaite compter. La fonction retourne le nombre de fois où la valeur v est présente sur la ligne li. L'exemple donné est nbValeurs(0, 1), qui renvoie 3.

On parcourt les éléments de la ligne d'indice `li`. On initialise un compteur à zéro et on l'incrémente chaque fois qu'un élément est égal à `v`. Le compteur obtenu est le nombre recherché.

RÉPONSE À RETENIR

```
def nbValeurs(li, v):
    compteur = 0
    for valeur in grille[li]:
        if valeur == v:
            compteur = compteur + 1
    return compteur
```

Pour l'exemple du sujet, `nbValeurs(0, 1)` renvoie 3.

b.

Question 2.b.

Sujet p. 10, 11

> Déduire une fonction regle1(li), dont le paramètre li indique le numéro de la ligne à remplir et qui modifie l'objet grille en remplissant de 0 ou de 1 à partir de la première règle. Dans l'exemple, la 6e ligne, initialement composée de deux cases blanches, puis 0, une case blanche, trois 0, une case blanche, 1 et 0, devient 1 1 0 1 0 0 0 1 1 0 après

FormaV - document d'entraînement non officiel

> l'appel regle1(6). S'il n'y a pas assez de 0 ou de 1, la fonction ne modifie rien.

Une ligne de 10 cases doit contenir cinq 0 et cinq 1. Si elle contient déjà cinq 0, chaque case encore blanche doit être un 1 ; si elle contient déjà cinq 1, chaque case blanche doit être un 0. Si aucune de ces conditions n'est satisfaite, la fonction ne modifie pas la ligne.

Pour l'exemple, la ligne donnée comporte cinq 0. Ses cases blanches sont donc remplacées par des 1, ce qui donne `[1, 1, 0, 1, 0, 0, 0, 1, 1, 0]`.

Le code suit l'appel explicite `regle1(6)` du sujet : en Python, cela désigne l'élément d'indice 6 de `grille` (les indices commencent à 0).

RÉPONSE À RETENIR

```
def regle1(li):
    if nbValeurs(li, 0) == 5:
        valeur_a_placer = 1
    elif nbValeurs(li, 1) == 5:
        valeur_a_placer = 0
    else:
        return

    for col in range(10):
        if grille[li][col] == -1:
            grille[li][col] = valeur_a_placer
```

Le `return` anticipé garantit que la fonction ne modifie rien si la ligne ne contient ni cinq 0 ni cinq 1.

Question 3.

Sujet p. 10, 11

> L'extrait de code recherche deux cases identiques séparées par une case blanche ; dans ce cas, la valeur de la case blanche peut être déterminée grâce à la troisième règle, qui interdit plus de deux 0 ou plus de deux 1 à la suite, horizontalement ou verticalement. Recopier et compléter les blancs de la fonction regle3(li), dont le paramètre li indique le numéro de la ligne étudiée et qui modifie grille en utilisant cette remarque. L'exemple montre deux zéros séparés par une case blanche, qui doit nécessairement contenir un 1.

Une ligne comporte 10 cases, d'indices 0 à 9. Pour consulter la case `col + 2`, il faut donc limiter `col` aux indices de 0 à 7, d'où `range(8)`. Si les deux extrémités sont égales et déjà connues, et que la case intermédiaire est blanche, cette dernière doit prendre la valeur opposée : sinon, on obtiendrait trois valeurs identiques consécutives. Le test exclut le cas où les deux extrémités valent `-1`, valeur qui code une case blanche.

RÉPONSE À RETENIR

```
def regle3(li):
    for col in range(8):
        if (grille[li][col] == grille[li][col+2]
                and grille[li][col] != -1
                and grille[li][col+1] == -1):
            grille[li][col+1] = autre(grille[li][col])
```

Par exemple, deux 0 séparés par une case blanche conduisent à placer un 1 dans cette case.

Question 4.

Sujet p. 11

> On suppose que toutes les cases blanches ont été complétées. Chaque ligne constituée de 0 et de 1 peut être considérée comme un nombre écrit en base 2. Écrire le code d'une fonction convert(L), dont le paramètre est une liste L de 10 bits (0 ou 1) et qui renvoie la valeur décimale correspondant à la ligne L. Par exemple, convert([0,0,0,0,0,1,1,1,1,1]) doit renvoyer 31.

On convertit la liste binaire de gauche à droite. À chaque bit, on double la valeur déjà obtenue puis on ajoute ce bit : c'est la méthode de Horner appliquée en base 2.

Pour la liste donnée, les valeurs successives après lecture des bits sont `0, 0, 0, 0, 0, 1, 3, 7, 15, 31`. La valeur décimale obtenue est donc 31.

RÉPONSE À RETENIR

```
def convert(L):
    valeur = 0
    for bit in L:
        valeur = 2 * valeur + bit
    return valeur
```

Ainsi, `convert([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])` renvoie `31`.

Question 5.

Sujet p. 11

> On considère que les valeurs obtenues à l'aide de la fonction convert ont été stockées dans une nouvelle liste V et que l'on souhaite s'assurer qu'il n'y a pas de doublon. Proposer en langage naturel l'algorithme d'une fonction qui prend en paramètre une liste d'entiers et renvoie un booléen (VRAI ou FAUX) indiquant si cette liste possède des doublons.

On compare chaque entier aux éléments qui le suivent. Si deux entiers égaux sont trouvés, la liste possède un doublon et la fonction renvoie VRAI. Si toutes les paires ont été comparées sans égalité, elle renvoie FAUX. Pour une liste de longueur `n`, cette méthode effectue au plus un nombre de comparaisons proportionnel à n^2.

RÉPONSE À RETENIR

Algorithme en langage naturel :

1. Pour chaque position de la liste, comparer l'entier qui s'y trouve à chaque entier situé après cette position.
2. Si deux entiers comparés sont égaux, renvoyer VRAI.
3. Si toutes les comparaisons sont terminées sans trouver d'égalité, renvoyer FAUX.

Sujet : https://www.sujetdebac.fr/annales-pdf/2022/spe-numerique-informatique-2022-mayotte-liban-2-sujet-officiel.pdf
© FormaV EI • Proposition de correction, distincte d'un corrigé officiel.

FormaV - document d'entraînement non 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.