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.

Numérique et sciences informatiques

Session 2025 • Métropole • Jour 2

Code sujet **25-NSIJ2ME1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Arbres binaires, codage de Shannon-Fano et programmation Python
12 réponses • sujet p. 2–6

Exercice 2 — Base de données relationnelle d’une ludothèque, SQL et programmation Python
9 réponses • sujet p. 7–10

Exercice 3 — Programmation Python, sécurisation des communications et réseaux
18 réponses • sujet p. 10–15

Les numéros, parties et pages renvoient au sujet original. Chaque réponse rappelle la consigne, explique la démarche et donne le résultat.

FormaV - document d'entraînement non officiel

Exercice 1

6 points

Arbres binaires, codage de Shannon-Fano et programmation Python

Partie A

Question 1.

Sujet p. 2

> Écrire le mot binaire qui sera utilisé pour encoder le caractère espace, représenté par le symbole _ dans l'arbre de la Figure 1.

Pour encoder un symbole, on suit depuis la racine les branches qui aboutissent à sa feuille et on concatène leurs étiquettes. Pour l'espace, on suit la branche droite étiquetée 0, puis la branche gauche étiquetée 1, puis la branche droite étiquetée 0.

> **RÉPONSE À RETENIR**
>
> Le code binaire de l'espace est **010**.

Question 2.

Sujet p. 2

> Déterminer le texte codé par le mot binaire 000111010111111100011001.

On lit le mot binaire de gauche à droite. Dès qu'un chemin aboutit à une feuille, on écrit le symbole correspondant et on reprend à la racine avec les bits suivants. Les codes utiles ici sont notamment e : 00, s : 011, p : 1010, i : 1111, o : 11001 et n : 1001.

Le découpage obtenu est `00 | 011 | 1010 | 1111 | 11001 | 1001`. Il correspond respectivement aux symboles e, s, p, i, o et n.

> **RÉPONSE À RETENIR**
>
> Le texte codé est **espion**.

Question 3.

Sujet p. 2, 3

> Citer le type de parcours de l'arbre qui permettrait d'obtenir les symboles classés par taille d'encodage croissante.

La taille d'encodage d'un symbole est la longueur du chemin de la racine à sa feuille. Un parcours en largeur visite les nœuds niveau par niveau ; les feuilles rencontrées sont donc classées par profondeur croissante, et donc par taille d'encodage croissante.

> **RÉPONSE À RETENIR**
>
> Il faut effectuer un **parcours en largeur** de l'arbre.

Partie B

Question 4.

Sujet p. 3

> Pour le texte « je pense, donc je suis », justifier par le calcul que l'étape 2 de l'algorithme de Shannon-Fano mène à

FormaV - document d'entraînement non officiel

> la situation illustrée par la Figure 2, à partir du tableau des symboles et de leurs nombres d'occurrences.

Le tableau est classé par occurrences croissantes et l'algorithme impose de séparer les symboles en deux groupes contigus dans ce classement. La somme de toutes les occurrences vaut :

$7 \times 1 + 2 + 2 + 3 + 4 + 4 = 22$. Dans le premier groupe de la Figure 2 — i, u, c, o, d, la virgule, p, n et j — la somme vaut $7 \times 1 + 2 + 2 = 11$. Dans le second groupe — s, l'espace et e — elle vaut $3 + 4 + 4 = 11$.

Les deux sommes sont égales : cette séparation atteint donc l'équilibre maximal possible.

> **RÉPONSE À RETENIR**
>
> La séparation de la Figure 2 est justifiée : les groupes ont chacun une somme d'occurrences égale à **11**, pour un total de **22**.

Question 5.

Sujet p. 4

> Donner la hauteur de l'arbre de la Figure 3 et préciser, dans le contexte de l'exercice, ce qu'elle représente. On rappelle qu'un arbre réduit à une seule feuille est de hauteur 0.

La hauteur est la longueur maximale, en nombre d'arêtes, d'un chemin allant de la racine à une feuille. Sur la Figure 3, le chemin le plus long aboutit à o ou d et compte 5 arêtes. Le code de ces feuilles a donc une longueur maximale de 5 bits.

> **RÉPONSE À RETENIR**
>
> La hauteur de l'arbre est **5**. Dans le contexte, elle représente la taille maximale, en bits, du code d'un symbole du texte.

Question 6.

Sujet p. 4

> En comparant le codage ASCII, où chaque symbole est codé sur un octet, et le codage de Shannon-Fano, justifier que ce second codage permet d'utiliser environ deux fois moins d'octets pour le texte « je pense, donc je suis ».

Le texte contient 22 caractères, espaces et virgule compris : en ASCII, il faut donc $22 \times 8 = 176$ bits, soit 22 octets.

Avec l'arbre de Shannon-Fano de la Figure 3, les longueurs des codes sont : i, u et c ont chacun 4 bits ; o et d ont chacun 5 bits ; la virgule et p ont chacun 4 bits ; n et j ont chacun 4 bits et apparaissent deux fois ; s apparaît trois fois avec un code de 3 bits ; l'espace apparaît quatre fois avec un code de 3 bits ; e apparaît quatre fois avec un code de 2 bits. Le nombre total de bits est donc $3 \times 4 + 2 \times 5 + 2 \times 4 + 2 \times (2 \times 4) + 3 \times 3 + 4 \times 3 + 4 \times 2 = 75$ bits.

Ces 75 bits représentent $75/8$, soit environ 9,4 octets ; en les stockant dans un nombre entier d'octets, il faut 10 octets. C'est environ 2,2 fois moins que les 22 octets ASCII.

> **RÉPONSE À RETENIR**
>
> Le codage de Shannon-Fano utilise **75 bits, soit 10 octets une fois stocké en octets entiers**, contre **22 octets** en ASCII : il utilise bien environ deux fois moins d'octets.

Question 7.

Sujet p. 3, 4

> Dessiner, en vous inspirant de l'arbre de la Figure 1, un arbre de codage qui permettrait d'encoder le mot « chiffrer

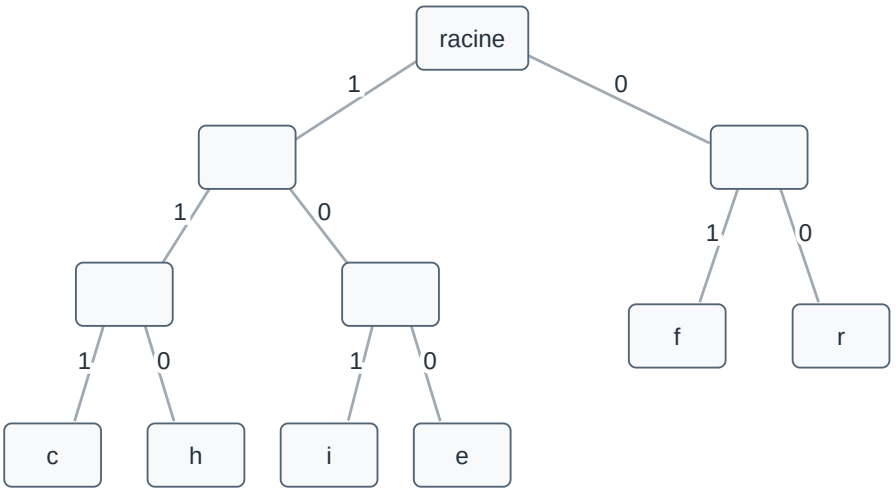
FormaV - document d'entraînement non officiel

» en utilisant l'algorithme de Shannon-Fano.

Dans « chiffrer », les occurrences sont c : 1, h : 1, i : 1, e : 1, f : 2 et r : 2. Un classement croissant compatible avec l'algorithme est c, h, i, e, f, r. La somme totale vaut 8. On peut séparer en deux groupes de somme 4 : c, h, i, e d'un côté et f, r de l'autre. Dans le premier groupe, on sépare c, h de i, e ; puis chaque paire est séparée en deux feuilles. Les branches gauches portent 1 et les branches droites 0.

On obtient ainsi c : 111, h : 110, i : 101, e : 100, f : 01 et r : 00. Les occurrences égales permettent d'autres classements et donc d'autres arbres valides.

Arbre de Shannon-Fano possible pour « chiffrer »



Les feuilles portent les symboles ; l'étiquette d'une branche est le bit ajouté au code.

RÉPONSE À RETENIR

Un arbre de codage possible est représenté ci-dessous. Les feuilles f et r ont un code de 2 bits ; les feuilles c, h, i et e ont un code de 3 bits.

Partie C

Question 8.

Sujet p. 5

Recopier et compléter les lignes 8 et 10 du code de la fonction creer_dico_occ.

Si le symbole est déjà dans le dictionnaire, son nombre d'occurrences augmente de 1. Sinon, sa première occurrence initialise ce nombre à 1.

RÉPONSE À RETENIR

```
8     dico[symbole] = dico[symbole] + 1
10    dico[symbole] = 1
```

Question 9.

Sujet p. 5

> Écrire une fonction somme_occ qui prend en paramètres un tableau tab de tuples (symbole, nb_occ) et qui renvoie la somme des nombres d'occurrences des symboles du tableau. Les tuples utilisés sont de même structure que l'élément renvoyé dans l'exemple.

Dans chaque tuple, le nombre d'occurrences se trouve à l'indice 1. On parcourt donc le tableau et on ajoute cette valeur à une somme initialisée à 0.

RÉPONSE À RETENIR

```
def somme_occ(tab):
    somme = 0
    for elt in tab:
        somme = somme + elt[1]
    return somme
```

Pour le tableau d'occurrences de l'exemple, la somme vaut $7 \times 1 + 2 + 2 + 3 + 4 + 4 = 22$.

Question 10.

Sujet p. 5, 6

> Recopier et compléter les lignes 9 et 11 du code de la fonction récursive shannon, qui prend en paramètres un caractère symbole et un tableau trié tab, et qui renvoie l'écriture binaire associée à symbole dans le tableau tab.

Si le symbole appartient au premier sous-tableau, son code commence par 1 et le reste de son code est calculé récursivement dans ce sous-tableau. Sinon, le code commence par 0 et la recherche récursive se fait dans le second sous-tableau.

RÉPONSE À RETENIR

```
9     return "1" + shannon(symbole, t1)
11    return "0" + shannon(symbole, t2)
```

Question 11.

Sujet p. 6

> Décrire ce qui garantit la terminaison de la fonction récursive shannon.

Lorsque le tableau contient un seul symbole, la fonction atteint son cas de base et renvoie la chaîne vide. Pour un tableau de plusieurs symboles, et avec des nombres d'occurrences positifs, la séparation donne deux sous-tableaux non vides. L'appel récursif se fait uniquement sur celui qui contient le symbole recherché : sa taille est strictement inférieure à celle du tableau initial. Comme le nombre de symboles est fini, les appels finissent par atteindre le cas de base.

RÉPONSE À RETENIR

La terminaison est garantie par la diminution stricte de la taille du tableau à chaque appel récursif et par le cas de base `len(tab) == 1`.

FormaV - document d'entraînement non officiel

Question 12.

Sujet p. 6

> Écrire une fonction encode_shannon qui prend en paramètre un texte de type str et renvoie un mot binaire de type str obtenu après encodage par l'algorithme de Shannon-Fano. On pourra utiliser les fonctions creer_dico_occ, creer_tab_trie, separe et shannon décrites dans le sujet.

On construit d'abord le dictionnaire des occurrences, puis le tableau trié. Pour chaque caractère du texte, on ajoute le code renvoyé par `shannon`. La chaîne vide est traitée à part afin de ne pas construire un tableau vide que la fonction `shannon` ne pourrait pas traiter.

La fonction `shannon` fournie renvoie la chaîne vide lorsque le tableau ne contient qu'un seul symbole. Le code ci-dessous respecte cette convention, mais un texte constitué d'un seul symbole répété ne peut alors pas être reconstitué à partir du mot binaire vide ; cette convention ne fournit donc pas, dans ce cas limite, un encodage sans perte.

RÉPONSE À RETENIR

```
def encode_shannon(texte):
    if texte == "":
        return ""
    dico = creer_dico_occ(texte)
    tab = creer_tab_trie(dico)
    mot_binaire = ""
    for symbole in texte:
        mot_binaire = mot_binaire + shannon(symbole, tab)
    return mot_binaire
```

Pour le texte `je pense, donc je suis`, chaque caractère est remplacé par son code dans le même tableau trié, puis les codes sont concaténés dans l'ordre du texte.

Exercice 2

6 points

Base de données relationnelle d'une ludothèque, SQL et programmation Python

Question 1.

Sujet p. 7, 8

> Expliquer pourquoi on ne peut pas prendre l'attribut nom comme clé primaire pour la relation adherent.

Une clé primaire doit identifier chaque ligne de manière unique. Or plusieurs adhérents peuvent avoir le même nom : l'attribut `nom` ne garantit donc pas l'unicité. Le schéma utilise à juste titre `idAdherent`, qui identifie chaque adhérent.

RÉPONSE À RETENIR

`nom` ne peut pas être clé primaire, car plusieurs adhérents peuvent porter le même nom.

Question 2.

Sujet p. 7, 8

> Décrire ce que donne la requête SQL suivante : SELECT nomJeu, editeur FROM jeu ORDER BY nomJeu;

`SELECT nomJeu, editeur` sélectionne les colonnes `nomJeu` et `editeur` de la table `jeu`. `ORDER BY nomJeu` classe les lignes par nom de jeu, par ordre croissant par défaut. La requête donne une ligne par jeu, avec son nom et son éditeur. Aucune donnée de jeu n'étant fournie, on ne peut pas énumérer les lignes obtenues.

RÉPONSE À RETENIR

La requête affiche, pour chaque jeu, son nom et son éditeur, les résultats étant triés par nom de jeu dans l'ordre croissant.

Question 3.

Sujet p. 7, 8

> Lorsque qu'un jeu est emprunté et n'a pas encore été rendu, la valeur de l'attribut dateRendu de la table emprunt est NULL. Écrire une requête permettant de connaître le nom de tous les jeux qui sont en cours d'emprunt.

Dans le schéma, `nomJeu` figure dans la table `emprunt`. Un emprunt en cours est caractérisé par un `dateRendu` égal à `NULL`. En SQL, on teste cette valeur avec `IS NULL`, et non avec `= NULL`. `DISTINCT` évite de répéter un nom si plusieurs lignes d'emprunt le fournissent.

RÉPONSE À RETENIR

```
SELECT DISTINCT nomJeu
FROM emprunt
WHERE dateRendu IS NULL;
```

Question 4.

Sujet p. 7, 8

> Écrire une requête SQL pour afficher le nom et le prénom de tous les adhérents qui ont emprunté le jeu « Catan ».

On relie chaque emprunt à l'adhérent concerné grâce à leur attribut commun `idAdherent`. On filtre ensuite sur le jeu `Catan`. `DISTINCT` évite de répéter le même nom et prénom si un adhérent a emprunté ce jeu plusieurs fois.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
SELECT DISTINCT adherent.nom, adherent.prenom
FROM adherent
JOIN emprunt ON adherent.idAdherent = emprunt.idAdherent
WHERE emprunt.nomJeu = 'Catan';
```

Question 5.

Sujet p. 7, 8

> Claire VOYANT, adhérente de longue date à cette ludothèque, a emprunté le jeu « Catan » et l'a rendu le 3 juin 2025. Lors de l'emprunt, la valeur de id_emprunt était 1538. Écrire une requête SQL qui a permis de mettre à jour la base de données afin qu'elle prenne en compte que ce jeu a été rendu. Toutes les dates de la base de données sont écrites au format 'AAAA-MM-JJ'.

La date du 3 juin 2025 s'écrit `2025-06-03` au format demandé. On met à jour la ligne identifiée par la clé primaire de l'emprunt, dont la valeur est 1538. Le schéma de la figure nomme cette colonne `idEmprunt`.

RÉPONSE À RETENIR

```
UPDATE emprunt
SET dateRendu = '2025-06-03'
WHERE idEmprunt = 1538;
```

Question 6.

Sujet p. 7, 8

> Écrire une requête SQL qui permet de trouver le nom et la catégorie de tous les jeux de la ludothèque sortis à partir de 2010 et dont l'âge minimum est strictement inférieur à 10 ans.

La date de sortie doit être supérieure ou égale à 2010 : « à partir de 2010 » inclut l'année 2010. L'âge minimum doit être strictement inférieur à 10. Les deux conditions doivent être satisfaites, donc elles sont reliées par `AND`.

RÉPONSE À RETENIR

```
SELECT nomJeu, categorie
FROM jeu
WHERE anneeSortie >= 2010 AND ageMinimum < 10;
```

Question 7.

Sujet p. 9

> Proposer les clés étrangères de la table participation en précisant le nom des attributs auxquels elles font référence.

Une participation relie un événement à un adhérent. Elle doit donc contenir une clé étrangère vers la clé primaire de `evenement`, qui est `nom`, et une autre vers la clé primaire de `adherent`, qui est `idAdherent`. On peut nommer ces attributs comme suit.

RÉPONSE À RETENIR

- `participation.nomEvenement` référence `evenement.nom` ;
- `participation.idAdherent` référence `adherent.idAdherent`.

Question 8.

Sujet p. 9, 10

> Écrire un script Python permettant de créer le dictionnaire dict_emprunts qui, à chaque jeu emprunté, associe le nombre de fois où il a été emprunté, à partir de la liste des jeux créée par le programme fourni.

On parcourt la liste `liste` obtenue dans le programme fourni. Pour chaque jeu, si la clé existe déjà, on augmente son compteur ; sinon, on crée la clé avec la valeur 1. Ainsi, après chaque élément traité, le compteur associé à un jeu correspond au nombre d'occurrences de ce jeu parmi les éléments déjà parcourus. Le parcours est en temps linéaire en la taille de `liste` en moyenne. Les valeurs de cette liste n'étant pas fournies, on ne peut pas déterminer ici le dictionnaire effectivement obtenu.

RÉPONSE À RETENIR

```
dict_emprunts = {}
for jeu in liste:
    if jeu in dict_emprunts:
        dict_emprunts[jeu] += 1
    else:
        dict_emprunts[jeu] = 1
```

Question 9.

Sujet p. 10

> Proposer un script Python permettant de générer le podium des jeux les plus souvent empruntés. Le podium est modélisé par une liste de trois listes et doit inclure les jeux ex æquo à la première, à la deuxième ou à la troisième place. Pour l'exemple du sujet, dict_emprunts contient Terraforming Mars : 25, Codenames : 22, Agricola : 18, Puerto Rico : 18, Caylus : 18, Dominion : 22 et Dixit : 12 ; le podium attendu regroupe Agricola, Puerto Rico et Caylus, puis Dominion et Codenames, puis Terraforming Mars.

Les rangs du podium correspondent aux trois valeurs distinctes de nombre d'emprunts les plus élevées : une égalité forme un même groupe, comme dans l'exemple. On récupère ces valeurs en ordre décroissant, puis on place le groupe le plus élevé dans la troisième liste, le deuxième dans la deuxième liste et le troisième dans la première liste, conformément à la représentation du sujet. L'ordre des jeux à l'intérieur d'un groupe d'ex æquo n'a pas d'importance.

Nombre d'emprunts	Jeux du groupe	Place représentée
25	Terraforming Mars	Première
22	Codenames, Dominion	Deuxième
18	Agricola, Puerto Rico, Caylus	Troisième
12	Dixit	Hors podium

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
valeurs = sorted(set(dict_emprunts.values()), reverse=True)[:3]
podium = [[], [], []]

for rang, nombre in enumerate(valeurs):
    groupe = [jeu for jeu, nb in dict_emprunts.items() if nb == nombre]
    podium[2 - rang] = groupe
```

Avec les données de l'exemple, le résultat est `[['Agricola', 'Puerto Rico', 'Caylus'], ['Codenames', 'Dominion'], ['Terraforming Mars']]`. Le groupe du milieu contient bien Dominion et Codenames ; leur ordre est interchangeable.

Exercice 3

8 points

Programmation Python, sécurisation des communications et réseaux

Partie A - La méthode du masque jetable

Question 1.

Sujet p. 10, 11

> Chiffrer, par la méthode du masque jetable, le message LIBRE à l'aide de la clé EYQMT.

On associe à chaque lettre son rang (A vaut 0, …, Z vaut 25), puis on additionne le rang de la lettre du message à celui de la lettre correspondante de la clé. On réduit chaque somme modulo 26.

Message	Rang	Clé	Rang	Somme réduite modulo 26	Lettre obtenue
L	11	E	4	15	P
I	8	Y	24	32 − 26 = 6	G
B	1	Q	16	17	R
R	17	M	12	29 − 26 = 3	D
E	4	T	19	23	X

RÉPONSE À RETENIR

Le chiffrement de `LIBRE` avec la clé `EYQMT` donne **PGRDX**.

Question 2.

Sujet p. 12

> Écrire une fonction Python indice qui prend pour paramètre une liste L et renvoie l'indice de element dans la liste L. On supposera que chaque élément de L n'y apparaît qu'une seule fois et que element est bien présent dans la liste L. Par exemple, indice(alphabet, 'K') renvoie l'entier 10.

On parcourt la liste en mémorisant chaque indice. Dès que l'élément à cet indice est égal à `element`, on renvoie l'indice. L'énoncé garantit que l'élément est présent dans la liste.

```
def indice(L, element):
    for i in range(len(L)):
        if L[i] == element:
            return i
```

RÉPONSE À RETENIR

Par exemple, `indice(alphabet, 'K')` renvoie **10**, car `'K'` est à l'indice 10 de la liste `alphabet`.

Question 3.

Sujet p. 12

> Écrire une fonction Python lettres_vers_indices qui prend pour paramètre une chaîne de caractères et renvoie, dans l'ordre, la liste des indices de ces caractères dans l'alphabet. Par exemple, lettres_vers_indices('HELLO')

FormaV - document d'entraînement non officiel

renvoie [7, 4, 11, 11, 14].

On parcourt la chaîne dans l'ordre et on cherche l'indice de chaque lettre dans `alphabet` avec la fonction `indice`. La liste construite conserve l'ordre des caractères de la chaîne.

```
def lettres_vers_indices(texte):
    return [indice(alphabet, lettre) for lettre in texte]
```

RÉPONSE À RETENIR

Ainsi, `lettres_vers_indices('HELLO')` renvoie `[7, 4, 11, 11, 14]`.

Question 4.

Sujet p. 12

Recopier et compléter les lignes 7 à 13 de la fonction chiffrement, qui renvoie la chaîne de caractères représentant le message chiffré par la méthode du masque jetable.

À chaque position `k`, on additionne les rangs du message et de la clé. Si la somme atteint au moins 26, on retranche 26 pour obtenir un rang entre 0 et 25. Une seule soustraction suffit puisque la somme maximale est 25 + 25 = 50. On convertit ensuite la liste des rangs en lettres.

RÉPONSE À RETENIR

```
    for k in range(n):
        ind = indices_msg[k] + indices_cle[k]
        if ind >= 26:
            ind = ind - 26
        indices_msg_chiffre.append(ind)
    msg_chiffre = indices_vers_lettres(indices_msg_chiffre)
    return msg_chiffre
```

Question 5.

Sujet p. 12, 13

Indiquer, en justifiant, ce que l'on observe lors de l'appel chiffrement('RESEAU', 'GFTZ').

La fonction commence par vérifier que la clé est au moins aussi longue que le message. Or `RESEAU` contient 6 caractères et `GFTZ` en contient 4. La condition de l'assertion est donc fausse et l'exécution s'arrête avant le chiffrement.

RÉPONSE À RETENIR

L'appel déclenche une `AssertionError` avec le message `impossible` : la clé est trop courte.

Question 6.

Sujet p. 11, 13

Déchiffrer le message GMEDH avec la clé FVEIT.

Pour chaque position, on soustrait le rang de la clé à celui de la lettre chiffrée. Si le résultat est négatif, on ajoute 26 afin de retrouver un rang de 0 à 25.

Lettre chiffrée	Rang	Clé	Rang	Calcul	Lettre déchiffrée
G	6	F	5	6 − 5 = 1	B
M	12	V	21	12 − 21 + 26 = 17	R
E	4	E	4	4 − 4 = 0	A
D	3	I	8	3 − 8 + 26 = 21	V
H	7	T	19	7 − 19 + 26 = 14	O

RÉPONSE À RETENIR

Le message `GMEDH` déchiffré avec la clé `FVEIT` est **BRAVO**.

Question 7.

Sujet p. 11, 13

> Expliquer comment procéder pour déchiffrer un message lorsqu'on connaît la clé.

On convertit le message chiffré et la clé en rangs. Pour chaque position, on soustrait le rang de la clé au rang du message chiffré, puis on réduit le résultat modulo 26. Concrètement, si la soustraction est négative, on ajoute 26. Enfin, on convertit les rangs obtenus en lettres.

RÉPONSE À RETENIR

Pour un rang chiffré `C` et un rang de clé `K`, le rang du message clair est `(C - K) modulo 26`.

Question 8.

Sujet p. 12, 13

> Adapter les lignes 6 à 13 de la fonction chiffrement pour obtenir la fonction dechiffrement. Les paramètres et les lignes 2 à 5 restent inchangés ; à la ligne 6, remplacer le nom de la variable indices_msg_chiffre par indices_msg_dechiffre.

Le déchiffrement inverse l'addition du chiffrement : on soustrait le rang de la clé au rang du message chiffré. Si le résultat est négatif, on ajoute 26. Voici la fonction complète, avec les lignes 2 à 5 conservées comme demandé :

RÉPONSE À RETENIR

```
def dechiffrement(msg, cle):
    assert len(cle) >= len(msg), 'impossible'
    indices_msg = lettres_vers_indices(msg)
    indices_cle = lettres_vers_indices(cle)
    n = len(msg)
    indices_msg_dechiffre = []
    for k in range(n):
        ind = indices_msg[k] - indices_cle[k]
        if ind < 0:
            ind = ind + 26
        indices_msg_dechiffre.append(ind)
    msg_dechiffre = indices_vers_lettres(indices_msg_dechiffre)
    return msg_dechiffre
```

Partie B - Sécurisation des communications

Question 9.

Sujet p. 13

> Expliquer la différence entre un algorithme de chiffrement symétrique et un algorithme de chiffrement asymétrique.

Avec un chiffrement symétrique, les correspondants utilisent la même clé secrète pour chiffrer et déchiffrer ; ils doivent donc partager cette clé de façon sûre. Avec un chiffrement asymétrique, chacun dispose d'une paire de clés liées : une clé publique, qui peut être diffusée, et une clé privée, qui doit rester secrète.

> **RÉPONSE À RETENIR**
>
> La différence essentielle est donc une clé secrète commune en chiffrement symétrique, contre une paire clé publique–clé privée en chiffrement asymétrique.

Question 10.

Sujet p. 13

> Bob envoie à Alice sa clé publique. Alice chiffre le message à l'aide de la clé publique de Bob puis le lui envoie. Indiquer comment Bob peut déchiffrer le message que lui envoie Alice.

Le message a été chiffré avec la clé publique de Bob. La clé permettant de le déchiffrer est la clé privée correspondante, que Bob garde secrète.

> **RÉPONSE À RETENIR**
>
> Bob déchiffre le message avec sa **clé privée**.

Question 11.

Sujet p. 13

> Expliquer comment une tierce personne pourrait se faire passer pour Alice sans que Bob ne s'en aperçoive.

Chiffrer avec la clé publique de Bob garantit que seul le détenteur de la clé privée correspondante peut déchiffrer le message, mais ne prouve pas l'identité de son expéditeur. Une tierce personne peut donc envoyer à Bob un message chiffré avec la clé publique de Bob et prétendre qu'Alice en est l'expéditrice. Sans signature ou autre mécanisme d'authentification, Bob ne peut pas vérifier cette identité.

> **RÉPONSE À RETENIR**
>
> Une tierce personne peut se faire passer pour Alice en envoyant à Bob un message chiffré avec la clé publique de Bob et en prétendant qu'Alice l'a envoyé.

Question 12.

Sujet p. 13

> Expliquer brièvement le fonctionnement du protocole HTTPS.

HTTPS est le protocole HTTP utilisé au-dessus de TLS. Lors de l'établissement de la connexion, le navigateur vérifie le certificat du serveur : il s'assure notamment que le nom du serveur correspond et que le certificat est lié à une clé publique par une chaîne d'autorités de certification auxquelles il fait confiance. TLS établit ensuite des clés de session ; les données sont généralement protégées par chiffrement symétrique et par un mécanisme garantissant leur intégrité.

RÉPONSE À RETENIR

Sous l'hypothèse que le certificat est correctement vérifié et que les clés restent secrètes, HTTPS protège la confidentialité et l'intégrité des échanges et authentifie le serveur. L'authentification du client n'est pas systématique.

Question 13.

Sujet p. 13

> Expliquer pourquoi, pour sécuriser intégralement les communications sur Internet, on utilise le protocole HTTPS plutôt qu'un chiffrement asymétrique.

Le chiffrement asymétrique est plus coûteux en calcul que le chiffrement symétrique et n'est donc pas adapté au chiffrement direct de grandes quantités de données. HTTPS utilise TLS pour authentifier le serveur et établir des clés de session, puis emploie généralement un chiffrement symétrique efficace pour les données. TLS ajoute aussi des protections d'intégrité : le chiffrement seul ne garantit pas nécessairement l'intégrité ni l'identité de l'expéditeur.

RÉPONSE À RETENIR

HTTPS combine donc des mécanismes asymétriques, utiles notamment à l'établissement de la connexion, avec un chiffrement symétrique plus efficace pour protéger les échanges de données.

Partie C - Réseaux

Question 14.

Sujet p. 14

> Marc, dont le poste de travail appartient au réseau 192.168.110.0/24 et dont l'identifiant hôte est 153, veut tester la communication avec Bob, dont l'identifiant hôte est 115. Il exécute ping 192.168.100.115 et obtient 4 packets transmitted, 0 received, 100% packet loss. Expliquer l'affichage obtenu et corriger l'erreur de Marc.

Dans le réseau indiqué, Bob a l'adresse `192.168.110.115`. Marc a saisi `192.168.100.115` : le troisième octet est `100` au lieu de `110`. Il ne teste donc pas l'adresse de Bob. L'affichage indique que les quatre paquets envoyés à l'adresse saisie n'ont reçu aucune réponse ; cela ne permet pas, à lui seul, de déterminer si cette adresse erronée correspond à une machine.

RÉPONSE À RETENIR

La commande correcte est `ping 192.168.110.115`.

Question 15.

Sujet p. 14

> Donner la représentation décimale du masque de sous-réseau 11111111.11111111.11111111.11100000.

Les trois premiers octets `11111111` valent chacun 255. Le dernier octet vaut `11100000 = 128 + 64 + 32 = 224`.

RÉPONSE À RETENIR

La représentation décimale du masque est **255.255.255.224**.

FormaV - document d'entraînement non officiel

Question 16.

Sujet p. 14, 15

> Indiquer le nombre total d'adresses IPv4 pouvant être attribuées sur le sous-réseau d'adresse 192.168.110.96, sur lequel se trouve Bob.

Le masque `255.255.255.224` correspond à `/27` : 27 bits identifient le réseau et il reste `32 − 27 = 5` bits pour la partie hôte. Le sous-réseau contient donc $2^5 = 32$ adresses, de `192.168.110.96` à `192.168.110.127`. Dans la convention IPv4 usuelle, l'adresse du réseau (.96) et l'adresse de diffusion (.127) ne sont pas attribuables à des postes ; il reste alors 30 adresses hôtes utilisables. L'énoncé page 14 présente le nombre total d'adresses du sous-réseau en comptant les 256 adresses d'un /24, ce qui conduit ici à compter 32 adresses au total.

> **RÉPONSE À RETENIR**
>
> Le sous-réseau contient **32 adresses IPv4 au total**. Parmi elles, **30** sont attribuables à des postes selon la convention IPv4 usuelle.

Question 17.

Sujet p. 15

> Donner la représentation binaire du nombre 134.

On décompose `134 = 128 + 4 + 2`. Dans l'écriture binaire sur un octet, les bits de poids 128, 4 et 2 valent 1 ; les autres valent 0.

> **RÉPONSE À RETENIR**
>
> La représentation binaire de 134 est **`10000110`**.

Question 18.

Sujet p. 14, 15

> Zoé, dont l'adresse IPv4 est 192.168.110.134, exécute deux commandes : commande n°1, ping 192.168.110.115 ; commande n°2, ping 192.168.110.153. Indiquer, en justifiant, laquelle a produit l'affichage 4 packets transmitted, 4 received, 0% packet loss.

Avec le masque `/27`, les sous-réseaux ont chacun 32 adresses. Zoé, à l'adresse `192.168.110.134`, appartient au sous-réseau `192.168.110.128/27` (adresses de .128 à .159). Marc, à l'adresse `192.168.110.153`, appartient au même sous-réseau. Bob, à l'adresse `192.168.110.115`, appartient au sous-réseau `192.168.110.96/27`. La commande n°2 vise donc Marc, qui est dans le même sous-réseau que Zoé. Toutefois, l'énoncé précise que les sous-réseaux sont reliés par des routeurs : les adresses seules ne permettent pas d'exclure que le ping vers Bob réussisse aussi, si le routage et les réponses ICMP sont autorisés.

> **RÉPONSE À RETENIR**
>
> La réponse attendue en se fondant sur l'appartenance au même sous-réseau est la **commande n°2 : `ping 192.168.110.153`**. Strictement, les informations données ne suffisent pas à déterminer de façon certaine laquelle a produit l'affichage : un ping vers Bob pourrait aussi réussir si le routage entre sous-réseaux et les réponses ICMP sont autorisés.

Sujet : https://www.sujetdebac.fr/annales-pdf/2025/spe-numerique-informatique-2025-metropole-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.