Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour consulter tous nos sujets d'annales et corrigés du Bac Général Maths + NSI, veuillez visiter :
bac-maths-nsi.fr/annales-sujets-et-corriges

Amérique du nord - mai 2026 - sujet 1 (corrigé)

Exercice 1 (POO et récursivité)
Partie A : grille de jeu et score associé

1. La méthode suivante convient :

```
def __init__(self):
    self.grille = [[0 for j in range(7)] for i in range(6)]
```

2. Le code suivant convient :

```
def joue(self, colonne, joueur):
    ligne = 5
    while ligne != -1 and self.grille[ligne][colonne] != 0:
        ligne = ligne - 1
    if ligne != -1:
        self.grille[ligne][colonne] = joueur
        return True
    else:
        return False
```

3. Le code suivant convient :

```
jeu1 = Grille()
jeu1.joue(2,1)
jeu1.joue(3,2)
jeu1.joue(3,1)
```

4. Le score est obtenu à l'aide de l'instruction `valeur_case(5,3) - valeur_case(5,2) - valeur_case(4,3)`
5. La méthode suivante convient :

```
def score(self):
    s = 0
    for ligne in range(6):
        for colonne in range(7):
            if self.grille[ligne][colonne] == 1:
                s = s - valeur_case(ligne,colonne)
            elif self.grille[ligne][colonne] == 2:
                s = s + valeur_case(ligne,colonne)
    return s
```

Partie B : algorithme min-max et création de l'arbre de coups

6. Le code suivant convient :

```
class Noeud:
    def __init__(self,colonne):
        self.colonne = colonne
        self.score = 0
        self.suivants = []
```

7. La méthode suivante convient :

```
def colonne_score_min(self):
    meilleur = self.suivants[0]
    for noeud in self.suivants:
        if noeud.score < meilleur.score:
            meilleur = noeud
    return (meilleur.colonne, meilleur.score)
```

8. Le code suivant convient :

```
def calcule_score(self, niveau, joueur, grille):
    g = grille.gagnant()
    if g == 1:
        self.score = -(100 + 10*(niveau_max-niveau))
    elif g == 2:
        self.score = 100 + 10*(niveau_max-niveau)
    elif niveau == niveau_max:
        self.score = grille.score()
    else:
        for colonne in range(7):
            grille2=grille.copie_grille()
            if grille2.joue(colonne, joueur):
                nouveau_noeud = Noeud(colonne)
                self.suivants.append(nouveau_noeud)
                nouveau_noeud.calcule_score(niveau+1, 3-joueur, grille2)
        if joueur == 1:
            self.score = self.colonne_score_min()[1]
        else:
            self.score = self.colonne_score_max()[1]
```

9. Chaque coup peut produire jusqu'à 7 nouveaux coups. Ainsi, l'arbre contient environ 7^{42} possibilités, ce qui est considérable. Le temps de calcul sera donc beaucoup trop long

Partie C : choix du meilleur coup à jouer

10. La fonction suivante convient :

```
def choisit_coup(grille,joueur):
    racine = Noeud(-1)
    racine.calcule_score(0, joueur, grille )
    if joueur == 1:
        return racine.colonne_score_min()[0]
    else:
        return racine.colonne_score_max()[0]
```

Exercice 2 (Réseaux, structures de données et POO)

1. Un réseau en `/29` contient $2^{32-29} = 2^3 = 8$ adresses possibles. Parmi ces adresses, deux ne sont pas attribuables : l'adresse réseau et l'adresse de diffusion. On ne peut donc attribuer que 6 adresses possibles, celles de `10.42.0.81` à `10.42.0.86`. Une adresse possible pour le routeur E est donc `10.42.0.81`.
2. Le réseau de la salle Gaming Online est `10.42.0.64/28`. Ce réseau contient 16 adresses de `10.42.0.64` à `10.42.0.79`. Ainsi, la machine d'adresse `10.42.0.70` appartient au réseau « Salle Gaming Online »
3. Un réseau en `/30` contient $2^{32-30} = 2^2 = 4$ adresses possibles et donc seulement deux adresses attribuables.
   - Pour le réseau `10.42.0.16/30`, ces deux adresses sont `10.42.0.17` et `10.42.0.18` qui correspondent respectivement au routeur C et au routeur F.
   - Pour le réseau `10.42.0.12/30`, ces deux adresses sont `10.42.0.13` et `10.42.0.14` qui correspondent respectivement au routeur C et au routeur D.
4. La table suivante est possible :

Routeur C		
Réseau	Passerelle	Nombre de sauts
Gaming Online	connecté	0
Gaming VR	`10.42.0.18`	1
DMZ	`10.42.0.1`	1
Internet	`10.42.0.1`	2
Administration	`10.42.0.1`	2
Application	`10.42.0.14`	1
SGBD	`10.42.0.14`	2

   Pour le réseau « Administration », on pourrait aussi passer par D puis E
5. - Les liaisons à 10 Gb/s ont un coût de $10^{10}/10^{10} = 1$
   - Les liaisons à 1 Gb/s ont un coût de $10^{10}/10^{9} = 10$.
   - Les liaisons à 100 Mb/s ont un coût de $10^{10}/10^{8} = 100$.
6. On a deux chemins possibles : A - B - C - D et A - B - E - D, avec un coût de 12
7. Les données d'un segment TCP sont contenues dans un paquet IP
8. Avec une file, le premier paquet arrivé est le premier transmis car cela respecte l'ordre d'arrivée. En revanche, avec une pile, c'est le dernier paquet arrivé qui est transmis en premier, ce qui retarde les paquets les plus anciens. Une file est donc plus adaptée
9. Le code suivant convient :

```
class Routeur_DROP_TAIL:
    def __init__(self, t_max):
        self.f = cree_file()
        self.t_max = t_max
        self.t = 0
```

10. Le code suivant convient :

```
def recoit(self, p):
    if self.t < self.t_max:
        enfile(self.f, p)
        self.t = self.t + 1
        return True
    return False
```

11. Le code suivant convient :

```
def recoit(self, p):
    if self.t < self.t_min:
        enfile(self.f, p)
        self.t = self.t + 1
        return True
    elif self.t_min <= self.t < self.t_max:
        if self.tirage_au_sort():
            enfile(self.f, p)
            self.t = self.t + 1
            return True
        return False
    return False
```

Exercice 3 (Récursivité, programmation dynamique et langage SQL)

Partie A

1. Le numéro d'un immeuble dans une rue n'est pas forcément unique dans toute la base, donc ne peut pas servir de clé primaire
2. La requête suivante convient :

```
SELECT id_immeuble FROM immeuble WHERE rue_immeuble = 'la mer' ORDER BY id_immeuble
```

3. La requête suivante convient :

```
SELECT id_appart FROM appartement WHERE id_immeuble = 16 AND etage_appart >= 5
```

4. Si on supprime l'immeuble sans supprimer ou modifier les appartements associés, certains appartements feront référence à un immeuble qui n'existe plus, ce qui rompt l'intégrité référentielle.
5. La requête suivante convient :

```
INSERT INTO immeuble VALUES (140, 6, 13, 'Turing')
```

6. La requête suivante convient :

```
UPDATE appartement SET prix_appart = 2 * prix_appart WHERE id_appart = 603
```

7. La requête suivante convient :

```
SELECT MAX(prix_appart) FROM appartement
JOIN immeuble ON appartement.id_immeuble = immeuble.id_immeuble
WHERE rue_immeuble = 'la mer'
```

Partie B

8. On a les sous-séquences suivantes : `[3, 8]`, `[3, 5]`, `[1, 8]`, `[1, 2]`, `[1, 5]` et `[2, 5]`
9. La plus longue sous-séquence est `[1, 2, 5]` qui est de longueur 3
10. La fonction suivante convient :

```
def est_strict_croissante(seq):
    for i in range(len(seq) - 1):
        if seq[i] >= seq[i + 1]:
            return False
    return True
```

11. Le code suivant convient :

```
def llsc_fin(tab, i):
    if i == 0:
        return 1
    max_len = 1
    for j in range(i):
        if tab[j] < tab[i]:
            max_len = max(max_len, llsc_fin(tab, j) + 1)
    return max_len
```

12. Le code suivant convient :

```
def llsc_dyn(tab):
    n = len(tab)
    dyn = [1] * n
    for i in range(1, n):
        for j in range(i):
            if tab[j] < tab[i]:
                dyn[i] = max(dyn[i], dyn[j] + 1)
    return max(dyn)
```

13. La programmation dynamique mémorise les résultats intermédiaires et évite les recalculs inutiles

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.