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

Métropole - juin 2026 - sujet 2 (corrigé)

Exercice 1 (Réseaux, routage et POO)

1. Les 16 premiers bits, c'est-à-dire les deux premiers octets, de l'adresse IP de PC C01 correspondent à l'adresse du réseau. L'adresse du réseau C est donc `172.16.0.0/16`
2. L'adresse de diffusion du réseau C est `172.16.255.255/16` (on met tous les bits des deux derniers octets à 1).
3. Il y a $2^{32-16} - 2 = 65534$ adresses IP disponibles pour des machines du réseau C.
4. On a la table de routage suivante :

Table de routage R1		
Destination	Passe par	Nombre de sauts
R2	R3	2
R3	R3	1
R4	R4	1
R5	R4	2

5. On a le chemin suivant : PC A01 - Switch 1 - R1 - R3 - R2 - Switch B - PC B01.
6. On a le nouveau chemin suivant : R1 - R4 - R5 - R2.
7. On a le tableau suivant :

Type de connexion	Débit (bit/s)	Coût
Ethernet	10^7	$10^8/10^7 = 10$
Fast Ethernet	10^8	$10^8/10^8 = 1$
Fibre	10^9	$10^8/10^9 = 0{,}1$

8. On a trois chemins possibles pour aller de R1 à R4 :
   - R1 - R4, de coût 10
   - R1 - R3 - R4, de coût $0{,}1 + 1 = 1{,}1$
   - R1 - R3 - R2 - R5 - R4, de coût $0{,}1 + 10 + 1 + 1 = 12{,}1$

   Le chemin choisi par le protocole OSPF est donc R1 - R3 - R4.
9. Dans la fonction `ajouter_route`, il n'y a aucun test de précondition fait pour vérifier si la taille de `liste_routes` est inférieure à `MAX_ROUTES`. Il faudrait par exemple rajouter, entre les lignes 4 et 5, l'instruction suivante :

```
assert len(liste_routes) < MAX_ROUTES, "Nombre maximum de routes atteint"
```

10. Le code suivant convient :

```
class Routage:
    def __init__(self, capacite = 5):
        self.capacite = capacite
        self.routes = []

    def ajouter(self, route):
        if len(self.routes) < self.capacite:
            self.routes.append(route)
        else:
            print("Nombre maximum de routes atteint")
```

11. La méthode suivante convient :

```
    def afficher(self):
        for route in self.routes:
            for r in route:
                print(r)
            print("---")
```

Exercice 2 (Mise au point de programme, gestion de bugs et graphes)
Partie A : mélange aléatoire

1. La valeur de `melangee[2][1]` est `10`
2. L'instruction suivante convient : `valeurs = [k for k in range(1, 17)]`
3. Le tri étant en place, il suffit de faire : `shuffle(valeurs)`
4. Le code suivant convient :

```
def en_grille(valeurs, n):
    assert len(valeurs) == n*n
    grille = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            grille[i][j] = valeurs[j*n+i]
    return grille
```

5. Le code suivant convient :

```
def en_grille(valeurs, n):
    assert len(valeurs) == n*n
    grille = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            grille[i][j] = valeurs[i*n+j] # inversion de i et j
    return grille
```

Partie B : grille résoluble

6. La liste aplatie est `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 15, 14]` et on observe que :
   - il y a trois couples de valeurs inversées : `(14, 15)`, `(15, 16)` et `(14, 16)`, c'est-à-dire trois inversions de couples d'indices `(13, 14)`, `(13, 15)` et `(14, 15)` ;
   - la distance à calculer vaut 2 (0 lignes et 2 colonnes).

   Ainsi, la somme du nombre d'inversions et de la distance vaut $3 + 2 = 5$, et donc la grille n'est pas résoluble.
7. Le test suivant convient : `assert compte_inversion([2, 1, 4, 3]) == 2`
8. Le code suivant convient :

```
def compte_inversion(valeurs):
    total = 0
    for i in range(len(valeurs)):
        for j in range(len(valeurs)):
            if i < j and valeurs[i] > valeurs[j]:
                total += 1
    return total
```

9. Le code suivant convient :

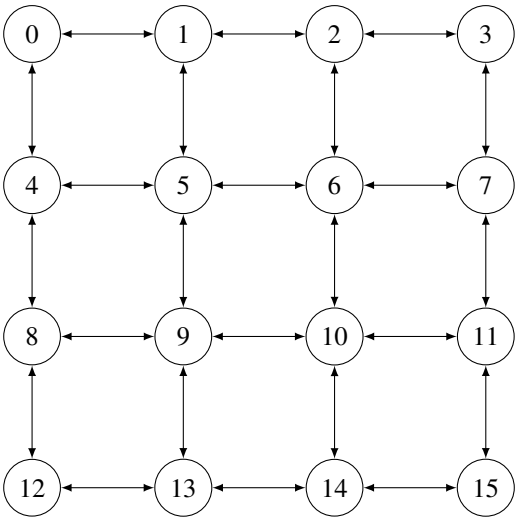
```
def distance_tuile_vide(grille, n):
    num_tuile_vide = n * n
    for i in range(n):
        for j in range(n):
            if grille[i][j] == num_tuile_vide:
                return n - 1 - i + n - 1 - j
```

10. Le code suivant convient :

```
def est_resoluble(valeurs, n):
    grille = en_grille(valeurs, n)
    inv = compte_inversion(valeurs)
    dis = distance_tuile_vide(grille, n)
    return (inv + dis) % 2 == 0
```

Partie C : mélange réaliste

11. Une valeur possible est `[5, 8, 10, 13]`. En effet, on a le graphe suivant :



12. Le code suivant convient :

```
def melange_graphe(nb_dep, n, graphe):
    valeurs = [i+1 for i in range(n * n)]
    num_tuile_vide = n * n
    actuelle = num_tuile_vide
    for k in range(nb_dep):
        prochaine = choice(graphe[actuelle])
        valeurs[actuelle] = valeurs[prochaine]
        valeurs[prochaine] = num_tuile_vide
        actuelle = prochaine
    return en_grille(valeurs, n)
```

Exercice 3 (Bases de données, files et graphe)

Partie A

1. Ces attibuts ne sont pas retenus comme clé primaire car on peut avoir des homonymes au sein de l'entreprise.
2. La relation `inscription` possède deux attributs, `trajet` et `passager` qui sont des clés étrangères, associées respectivement aux clés primaires `id_trajet` et `id_utilisateur` des relations `trajet` et `utilisateur`. Les clés étrangères ayant le même domaine que les clés primaires auxquelles elles font référence, leurs valeurs sont donc également des nombres entiers.
3. La requête suivante convient :

```
SELECT COUNT(*) FROM inscription WHERE trajet = 1291
```

4. La requête suivante convient :

```
SELECT * FROM trajet
WHERE heure >= '2026-06-19 00:00:00'
  AND heure < '2026-06-20 00:00:00'
  AND arrivee = 'Nancy'
ORDER BY heure
```

5. La requête suivante convient :

```
INSERT INTO trajet VALUES(1295, 'Nancy', 'Allain', '2026-06-19 17:45:00', 3, 25)
```

6. La requête suivante convient :

```
UPDATE trajet SET heure = '2026-06-19 18:40:00' WHERE id_trajet = 1294
```

7. L'erreur vient sans doute du fait que le trajet 1296 possède déjà des inscrits : il est donc référencé par la relation inscription, et on ne peut pas supprimer un trajet qui est référencé par une clé étrangère. Il faudrait d'abord supprimer les inscriptions associées à ce trajet avant de pouvoir le supprimer.
8. La requête suivante convient :

```
SELECT nom, prenom FROM utilisateur
JOIN inscription ON utilisateur.id_utilisateur = passager
JOIN trajet ON id_trajet = trajet
WHERE conducteur = 25
```

Partie B

9. Le dictionnaire suivant convient :

```
graphe = {'V': ['M'],
          'M': ['V', 'P2', 'P3'],
          'P2': ['M', 'A', 'C'],
          'P3': ['M', 'A'],
          'A': ['P2', 'P3', 'P1'],
          'P1': ['A', 'C'],
          'C': ['P1', 'P2']}
```

10. Le principe de fonctionnement d'une file est de stocker les éléments dans l'ordre dans lequel ils sont ajoutés, et de les retirer dans le même ordre, en suivant le principe FIFO (*First In, First Out*). Ainsi, le premier élément ajouté à la file sera le premier à être retiré.
11. Un ordre possible est le suivant : P1 - A - C - P2 - P3 - M - V.
12. Il s'agit d'un parcours en largeur.
13. Le code suivant convient :

```
def excentricite(graphe, sommet):
    dico = dico_distance(graphe, sommet)
    m = 0
    for distance in dico.values():
        if distance > m:
            m = distance
    return m
```

14. Le meilleur point de rendez-vous est le sommet P2, car il a l'excentricité la plus faible, qui est 2, parmi les points de rendez-vous. En effet, l'excentricité de P1 est 4 et celle de P3 est 3.

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.