Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

SUJET D'EXAMEN (SESSION OFFICIELLE)

Numérique et sciences informatiques - Session 2024

Sujet de l'examen national tel que publié pour la session indiquée. Source : sujets nationaux publiés par le ministère chargé de l'examen et les services académiques. Document mis à disposition par FormaV pour la révision.

Toutes les annales et leurs corrigés : https://bac-maths-nsi.fr/annales-sujets-et-corriges

BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2024

NUMÉRIQUE ET SCIENCES INFORMATIQUES

ÉPREUVE DU MERCREDI 11 SEPTEMBRE 2024

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice n'est pas autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 16 pages numérotées de 1/16 à 16/16.

L'annexe page 16 est à rendre avec la copie

Le sujet est composé de trois exercices indépendants.

Le candidat traite les trois exercices.

EXERCICE 1 (6 points)

Cet exercice porte sur la programmation Python, les bases de données relationnelles et les requêtes SQL.

En particulier, les mots-clés suivants peuvent être utilisés :

`SELECT`, `CREATE TABLE`, `FROM`, `WHERE`, `JOIN ON`, `INSERT INTO`, `VALUES`, `UPDATE`, `SET`, `COUNT`, `DELETE`, `DISTINCT`, `AND`, `OR`, `AS`.

Partie A

Dans cette partie, on utilise une base de données relationnelle.

Une entreprise de location de voitures propose à ses clients la possibilité de rapporter le véhicule dans une autre agence. Les informations correspondantes sont rangées dans une base de données. Voici les extraits de deux tables utilisées.

Agences					
Agence	Ville	CP	Departement	Adresse	Telephone
Licorne	Paris	75001	Paris	123 Rue de la Révolution	0123456789
Le Carosse	Paris	75001	Paris	456 Virtual Street	0156789012
Vroum	Lyon	69001	Rhône	789 Virtual Lane	0456789034
Rapide	Toulouse	31000	Haute Garonne	321 Virtual Avenue	0567890123
Deep Place	Bordeaux	33000	Gironde	987 Virtual Road	0567890145

Voitures						
id_voiture	marque	modele	kilometrage	nombre_places	type	carburant
1	Renault	Clio	64022	5	Berline	Essence
2	Renault	Clio	50350	5	Berline	Essence
3	Dacia	Sandero	62031	5	Berline	Essence
4	Dacia	Sandero	58955	5	Berline	Essence
5	Dacia	Sandero	65779	5	Berline	Essence
6	Dacia	Sandero	56253	5	Berline	Essence
7	Renault	Clio	49660	5	Berline	Essence
8	Fiat	500	2545	4	Citadine	Electrique
9	Fiat	500	1953	4	Citadine	Electrique
10	Fiat	500	549	4	Citadine	Electrique

1. Donner pour la table `Agences` un type possible pour l'attribut `CP` qui indique le code postal.

La fonction `COUNT` permet de compter le nombre d'enregistrements et le mot-clé `DISTINCT` permet de ne pas prendre en compte les doublons.

2. Donner le résultat de la requête suivante pour les extraits des tables données :

```
SELECT COUNT(DISTINCT 'Telephone')
FROM 'Agences';
```

3. Expliquer à quelle condition l'attribut `Telephone` pourrait servir de clé primaire pour la table `Agences`.

Certaines agences ont décidé de partager un même standard téléphonique. On ajoute à la table `Agences` l'attribut `id_agence` qui est utilisé comme clé primaire. Voici la nouvelle table obtenue :

Agences						
id_agence	Agence	Ville	CP	Departement	Adresse	Telephone
1	Licorne	Paris	75001	Paris	123 Rue de la Révolution	0123456789
2	Le Carosse	Paris	75001	Paris	456 Virtual Street	0123456789
3	Vroum	Lyon	69001	Rhône	789 Virtual Lane	0456789034
4	Rapide	Toulouse	31000	Haute Garonne	321 Virtual Avenue	0567890123
5	Deep Place	Bordeaux	33000	Gironde	987 Virtual Road	0567890145

Nous souhaitons créer une table `couple_voitures_agences` qui permettra d'associer la table `Agences` à la table `Voitures`. On précise que la table `Voitures` a pour clé primaire `id_voiture`.

4. Décrire les attributs de cette nouvelle table ; une clé primaire sera soulignée et une clé étrangère commencera par un dièse (#).

5. Écrire une requête qui permet d'enregistrer dans la table `couple_voitures_agences` le fait que le véhicule `2` est associé à l'agence `Deep Place`.

6. Écrire une requête qui permet d'actualiser la table pour indiquer que le véhicule `2` se trouve maintenant à l'agence `Le Carrosse`.

7. Écrire une requête qui permet d'afficher le type, la marque et le nom de l'agence pour l'ensemble des véhicules.

Partie B

Pour cette seconde partie, on admet que pour chacune des trois tables, la clé primaire est auto-incrémentée. On utilise par la suite la programmation en Python.

On dispose de la fonction `execute_requete_insert` :

```
 1  def execute_requete_insert(requete):
 2      """
 3      Exécute la requête d'insertion passée en paramètre.
 4      Paramètre :
 5      - requete : STR, la chaîne formatée de la requête
 6      Résultat :
 7      - BOOL, booléen de contrôle : Vrai si la requête s'est bien
passée, Faux sinon
 8      """
       ...
```

Exemple d'utilisation :

```
>>> execute_requete_insert('INSERT INTO couple_voitures_agences (id,
id_agence, id_voiture) VALUES (9, 3, 3)')
>>> True
```

On souhaite disposer d'une fonction Python qui insère un nouveau véhicule dans une agence. Cette fonction prend en paramètres les valeurs des attributs de la voiture à insérer et l'entier `id_agence`. Elle renvoie un booléen de contrôle.

```
   def insert_voiture(liste_valeurs, id_agence):
       ...
```

Exemple d'utilisation pour une Fiat dans l'agence Vroum :

```
>>> insert_voiture((‘Fiat’, ’500’, 18, 4, ’Citadine’,
‘Electrique’),3)

>>> True
```

8. Écrire en Python la fonction `insert_voiture`.

9. Préciser les conditions que doivent vérifier les paramètres transmis à cette fonction.

EXERCICE 2 (6 points)

Cet exercice porte sur les réseaux, les protocoles de routage et les graphes.

Partie A

Le réseau informatique d'une société est constitué d'un ensemble de routeurs interconnectés à l'aide de fibres optiques.

La figure ci-dessous représente le schéma de ce réseau. Il est composé de deux réseaux locaux L1 et L2. Le réseau local L1 est relié au routeur R1 et le réseau local L2 est relié au routeur R9.

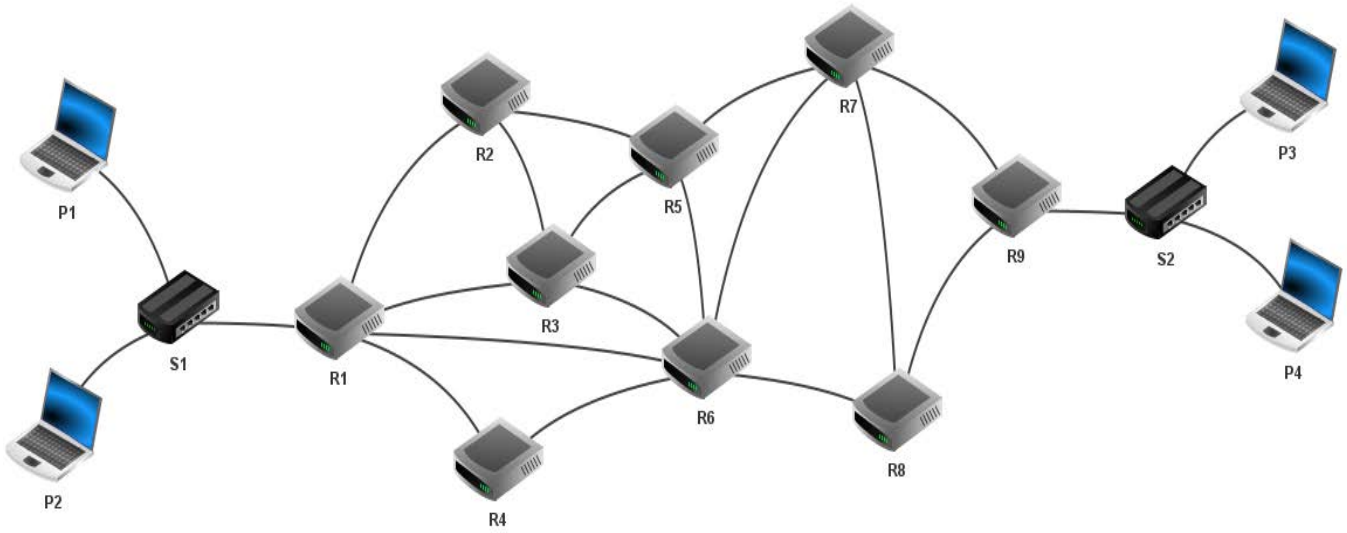


Figure 1. Réseau

Dans cette partie, les adresses IP sont composées de 4 octets, soit 32 bits. Elles sont notées X1.X2.X3.X4, où X1, X2, X3 et X4 sont les valeurs des 4 octets, converties en notation décimale. La notation X1.X2.X3.X4/n signifie que les n premiers bits de poids forts de l'adresse IP représentent la partie « réseau », les bits suivants représentent la partie « hôte ».

Toutes les adresses des machines connectées à un réseau local ont la même partie réseau.

Le tableau suivant indique les adresses IPv4 des machines constituant le réseau de la société.

NOM	TYPE	ADRESSE IPV4
R1	Routeur	Interface 1 :192.168.1.1/24 Interface 2 :192.168.2.1/24 Interface 3 :192.168.3.1/24 Interface 4 :192.168.4.1/24 Interface 5 :192.168.5.1/24

NOM	TYPE	ADRESSE IPV4
R2	Routeur	Interface 1 :192.168.2.2/24 Interface 2 :192.168.7.1/24 Interface 3 :192.168.8.1/24
R3	Routeur	Interface 1 :192.168.3.2/24 Interface 2 :192.168.7.2/24 Interface 3 :192.168.9.1/24 Interface 4 :192.168.10.1/24
R4	Routeur	Interface 1 :192.168.5.2/24 Interface 2 :192.168.6.1/24
R5	Routeur	Interface 1 :192.168.8.2/24 Interface 2 :192.168.9.2/24 Interface 3 :192.168.11.1/24 Interface 4 :192.168.12.1/24
R6	Routeur	Interface 1 :192.168.4.2/24 Interface 2 :192.168.6.2/24 Interface 3 :192.168.10.2/24 Interface 4 :192.168.11.2/24 Interface 5 :192.168.13.1/24 Interface 6 :192.168.14.1/24
R7	Routeur	Interface 1 :192.168.12.2/24 Interface 2 :192.168.13.2/24 Interface 3 :192.168.15.1/24 Interface 4 :192.168.16.1/24
R8	Routeur	Interface 1 :192.168.14.2/24 Interface 2 :192.168.15.2/24 Interface 3 :192.168.17.1/24
R9	Routeur	Interface 1 :192.168.16.2/24 Interface 2 :192.168.17.2/24 Interface 3 :192.168.18.1/24
P1	Portable	192.168.1.10
P2	Portable	Non fourni
P3	Portable	Non fourni
P4	Portable	Non fourni

1. En utilisant les adresses IP des différentes interfaces et des ordinateurs portables, en déduire une adresse possible pour le portable P2.

2. Donner l'adresse du réseau local L2 ainsi que le nombre d'adresses possibles pour les ordinateurs portables P3 et P4.

Partie B

Le graphe G, représenté ci-dessous, schématise l'architecture du réseau de la société. Les sommets représentent les routeurs et les arêtes représentent les liaisons.

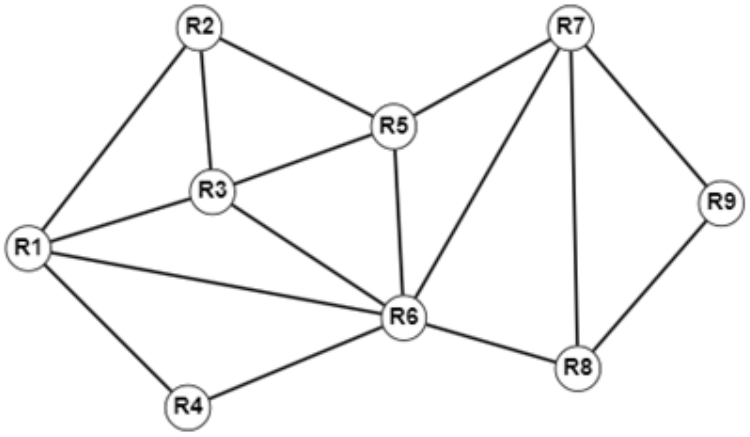


Figure 2. Graphe non pondéré

3. Donner l'implémentation Python des listes d'adjacence de ce graphe à l'aide d'un dictionnaire dont les clés sont les sommets et les valeurs la liste des sommets adjacents du sommet clé. On nomme G ce dictionnaire.

   Afin de faciliter la notation, on s'autorise à écrire chaque couple clé/valeur sur une nouvelle ligne.

On suppose que le protocole de routage RIP est utilisé.

4. Recopier et compléter, en rajoutant autant de lignes que nécessaire, la table de routage simplifiée suivante du routeur R1.

Destination	Suivant	Nombre de sauts
R2	R2	1
R3		

L'ordinateur P1 envoie un paquet de données à l'ordinateur P3.

5. Donner l'un des chemins empruntés par le paquet ainsi que le nombre de sauts.

La société doit vérifier l'état physique de la fibre optique installée sur le réseau. Un robot inspecte toute la longueur de la fibre optique afin de s'assurer qu'elle ne présente pas de détérioration apparente.

On appelle M la matrice d'adjacence du graphe de la figure 2. Les sommets sont rangés par ordre croissant des numéros des routeurs (R1, R2, …, R9).

6. Donner l'écriture en Python de cette matrice d'adjacence sous la forme d'une liste de listes.

Le degré d'un sommet est le nombre d'arêtes dont ce sommet est une extrémité.

7. Recopier et compléter les lignes 3, 5 et 6 de le fonction `degre` qui prend en paramètre la matrice d'adjacence d'un graphe donné sous forme d'une liste de listes et qui renvoie la liste des degrés de tous les sommets du graphe rangés dans le même ordre que les sommets de la matrice d'adjacence.

```
1 def degre(MATRICE):
2     d = []
3     for ... in ...:
4         cpt = 0
5         for ... in ...:
6             cpt = cpt +...
7         d.append(cpt)
8     return d
```

8. Donner la liste renvoyée par `degre(M)`.

On appelle chaîne eulérienne d'un graphe non orienté un chemin qui passe une et une seule fois par toutes les arêtes du graphe. Un graphe connexe admet une chaîne eulérienne si et seulement si le graphe possède, au plus, deux sommets de degré impair.

9. En utilisant le résultat de la question précédente et en admettant que le graphe est connexe, indiquer si le robot peut parcourir l'ensemble du réseau en suivant les fibres optiques et en empruntant chaque fibre optique une et une seule fois.

Partie C

Le poids sur chaque arête représente la bande passante en megabits par seconde (Mb/s) de chaque liaison.

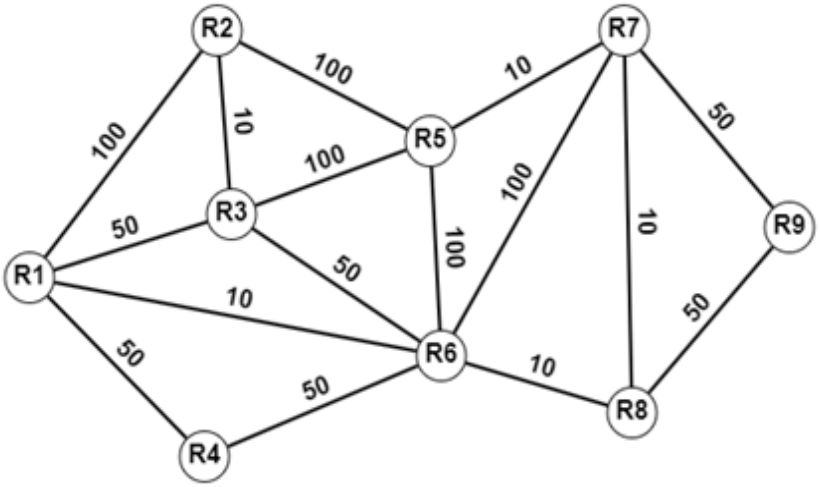


Figure 3. Graphe pondéré

Dans cette partie, on utilise le protocole de routage OSPF. Pour calculer le coût d'une liaison, on utilise la formule :

$$C = \frac{10^8}{BP}$$

où BP est la bande passante en bits par seconde.

10. Déterminer la route qui sera empruntée par le paquet pour aller de l'ordinateur P1 à l'ordinateur P3. Préciser le coût de ce trajet.

EXERCICE 3 (8 points)

Cet exercice porte sur les bases de numération, la structure de données PILE et la POO.

La civilisation *Maya* est une ancienne civilisation de Mésoamérique principalement connue pour ses avancées dans les domaines de l'écriture, de l'art, de l'architecture, de l'agriculture, des mathématiques et de l'astronomie.

La numération *Maya* est une numération positionnelle de base 20 (dite vigésimale) utilisant trois symboles pour former les “chiffres” :

- une coquille pour le zéro  ,
- un point pour l’unité ●,
- un trait pour la valeur 5 ▬.

Les “chiffres” sont les suivants. Ils utilisent une numération additive :

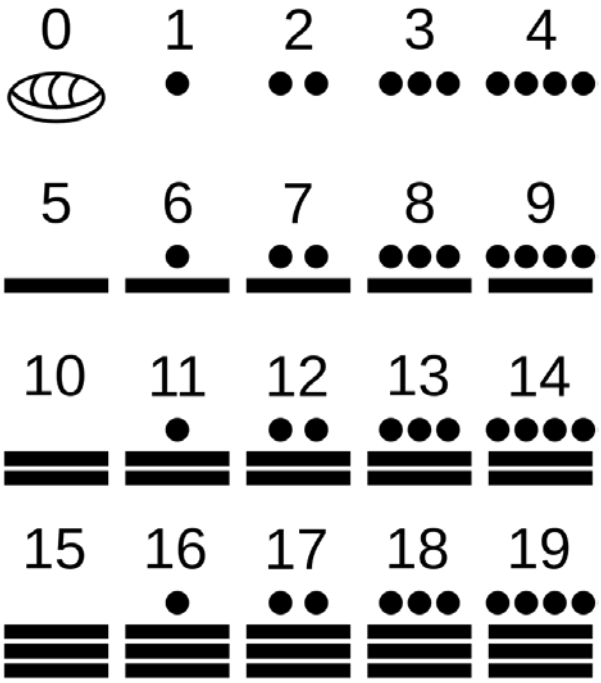
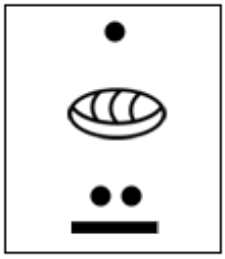


Figure 1. Table des “chiffres” et valeurs correspondantes (source : Wikipédia)

Dans une version simplifiée de ce système, l’écriture d’un nombre se fait par empilement de “chiffres”. Chaque étage correspond à un chiffre de poids 20 fois supérieur au poids du chiffre de l’étage inférieur.

Ainsi la valeur du chiffre de l’étage le plus bas est multipliée par 20^0 soit 1, du second étage par 20^1, du troisième étage par 20^2, et ainsi de suite.

Exemple : la représentation *Maya* de l'entier 407 est la suivante.



1. Compléter le tableau donné en **annexe à rendre avec la copie**.

2. Justifier que l'écriture *Maya* de l'entier 3435 est :

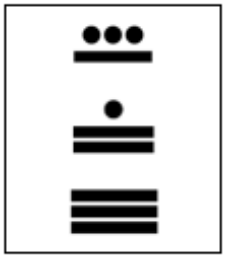


Figure 2. Ecriture *Maya* de l'entier 3435

On **modélise l'écriture d'un entier** dans sa représentation *Maya* par une pile formée de listes. Chacune de ces listes est composée de trois entiers et modélise le "chiffre" d'un étage :

- le premier entier vaut 0 ou 1 suivant s'il y a ou non une coquille ;
- le deuxième représente le nombre de points ;
- le troisième représente le nombre de traits.

Ainsi, la modélisation *Maya* de l'entier 3435 est `[[0, 0, 3], [0, 1, 2], [0, 3, 1]]` et celle de l'entier 407 est `[[0, 2, 1], [1, 0, 0], [0, 1, 0]]`.

Le sommet de la pile se situe en fin de liste.

On dispose de la classe suivante :

```
class Maya:
    def __init__(self):
        self.nombre = []

    def ajouter(self, chiffre):
        """ chiffre est une liste de longueur 3.
        La méthode empile le chiffre au sommet de la pile """
        self.nombre.append(chiffre)

    def retirer(self):
        """ depile et renvoie le chiffre qui etait au sommet de

          la pile """
        if not self.estVide():
            return self.nombre.pop()

    def estVide(self):
        return self.nombre == []

    def nbEtages(self) :
        """ renvoie le nombre de chiffres de la pile """
        ...

    def MayaToDec(self):
        """ renvoie le nombre entier correspondant a la
        modelisation Maya de l'instance courante """
        coeff = 20**...
        ch_Dec = 0
        while ... :
            ch_Maya = ...
            ch_Dec = ch_Dec + (valeurChiffre(ch_Maya)) * coeff
            coeff = ...
        return ch_Dec

    def multiplie(self):
        """ renvoie le resultat de la multiplication par 20 d'un
        nombre en modelisation Maya. """
        ...

    def somme(self, maya2):
        """ ajoute maya2 à l'instance courante et renvoie le
        resultat en modelisation Maya """
        if self.nbEtages() == maya2.nbEtages()
        ...
```

3. Écrire une suite d'instructions permettant de créer une instance, nommée `M`, de la classe `Maya` qui modélise le nombre entier 3435.

4. Écrire la méthode `nbEtages` de la classe `Maya`. Celle-ci renvoie le nombre de "chiffres" utilisés pour écrire le nombre correspondant en écriture *Maya*.

De l'écriture *Maya* à l'écriture décimale

5. Écrire une fonction `valeurChiffre` ayant pour paramètre une liste `L`. Celle-ci renvoie la valeur de l'entier associé à la liste `L = [c, p, t]` où `c` (de valeur 0 ou 1) indique la présence d'une coquille, `p` est le nombre de points et `t` le nombre de traits composant un "chiffre" *Maya*.

Exemple :

```
>>> valeurChiffre([0, 2, 3])
>>> 17
>>> valeurChiffre([1, 0, 0])
>>> 0
```

6. Recopier et compléter les lignes 2, 4, 5 et 7 de la méthode `MayaToDec` suivante de la classe `Maya`. Cette méthode renvoie la valeur de l'entier associé à l'objet `Maya`. On pourra utiliser les méthodes `estVide`, `nbEtages` et `retirer`.

```
1 def MayaToDec(self):
2     coeff = 20**...
3     ch_Dec = 0
4     while ... :
5         ch_Maya = ...
6         ch_Dec = ch_Dec + (valeurChiffre(ch_Maya)) * coeff
7         coeff = ...
8     return ch_Dec
```

De l'écriture décimale vers sa modélisation *Maya*

On considère que la fonction `DecToVige` est déjà écrite. Celle-ci prend en paramètre un entier `n` et renvoie la décomposition en base 20 de celui-ci sous la forme d'une liste $[\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_p]$ telle que :

$$n = \mathbf{a}_0 \times 20^0 + \mathbf{a}_1 \times 20^1 + \mathbf{a}_2 \times 20^2 + \ \ldots + \mathbf{a}_p \times 20^p$$

Exemple :

```
>>> DecToVige(3435)
[15, 11, 8]
>>> DecToVige(407)
[7, 0, 1]
```

7. Écrire la fonction `decompChiffre` qui prend en paramètre un entier `n` compris entre 0 et 19 et renvoie la liste `[c, p, t]` où `c` vaut 0 ou 1 et indique la présence ou non d'une coquille, `p` est le nombre de points et `t` le nombre de traits composant le "chiffre" *Maya* correspondant.

Exemple :

```
>>>decompChiffre(17)
[0, 2, 3]
```

```
>>>decompChiffre(0)
[1, 0, 0]
```

8. Écrire la fonction `DecToMaya` qui prend en paramètre un entier `n` et renvoie la modélisation *Maya* d'un objet M de la classe `Maya` correspondant.

Exemple :

```
>>>DecToMaya(3435).nombre
[[0, 0, 3], [0, 1, 2], [0, 3, 1]]
>>>DecToMaya(407).nombre
[[0, 2, 1], [1, 0, 0], [0, 1, 0]]
```

Opérations sur les nombres en modélisation *Maya*

On souhaite additionner des nombres directement à partir de leur modélisation *Maya*.

9. Écrire la méthode `multiplie` de la classe `Maya` qui renvoie le résultat de la multiplication par 20 d'un nombre en modélisation *Maya*.

Exemple :

```
>>> M = Maya()
>>> M.ajouter([0 ,0, 3])
>>> M.ajouter([0, 1, 2])
>>> M.multiplie().nombre
[[1, 0, 0], [0, 0, 3], [0, 1, 2]]
```

On donne la fonction `mystere` suivante :

```
1 def mystere(m1, m2, ret):
2     c = 0
3     p = (m1[1] + m2[1] + ret)%5
4         if m1[1] + m2[1] + ret >= 5:
5             ret = 1
6         else:
7             ret = 0
8     t = (m1[2] + m2[2] + ret)%4
9     if m1[2] + m2[2] + ret < 4:
10        ret = 0
11    else:
12        ret = 1
13    if (m1[0] == 0 and m2[0] == 1) or (p + t = 0 and ret == 1):
14        c = 1
15    return ([c, p, t], ret)
```

10. Donner les résultats renvoyés par les deux appels suivants :

- `mystere([0, 1, 1], [0, 3, 1], 0)`
- `mystere([0, 1, 1], [0, 4, 2], 0)`

11. Écrire une méthode `somme` de la classe `Maya` permettant d'ajouter à l'instance courante un autre nombre `maya2` de même taille en modélisation Maya.

ANNEXE À RENDRE AVEC LA COPIE

Exercice 3 – Question 1.

Étage	Écriture *Maya*	Valeur du “chiffre” de l’étage	Valeur dans la conversion
3		$1 \times 5 + 3 \times 1 = 8$	$8 \times 20^2 = 3200$
2			
1			

Modèle CCYC : ©DNE

NOM DE FAMILLE (naissance) **:**
(en majuscules)

PRENOM :
(en majuscules)

N° candidat :

N° d'inscription :

(Les numéros figurent sur la convocation, si besoin demander à un surveillant.)

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

Né(e) le : / /

1.2