

Nom :

Prénom :

Groupe :

Le sujet comporte 5 exercices et 5 pages. *Aucun document n'est autorisé* : les principales fonctions de manipulation des graphes sont rappelées en annexe, page 6.

Exercice 1 Voici une fonction Python :

```
def mystere(a,n):  
    u = a  
    for k in range (n):  
        u = 2*u + 1  
    return u
```

1. Simuler l'exécution de `mystere(1,6)` en complétant le tableau suivant :

| | |
|-----|-----|
| k | 0 |
| u | 1 3 |

2. Que vaut `mystere(1,6)` ?

3. Que vaut `mystere(0,6)` ?

4. Pour tout couple d'entiers positifs ou nuls (a, n) on note $f(a, n)$ la valeur de `mystere(a,n)` ; écrire une fonction Python `mystere2(a,x)` qui calcule le plus petit entier n tel que $f(a, n) > x$:

5. Donner une formule pour $f(1, n)$:

(on ne demande pas de preuve).

Exercice 2 Pour tester si deux sommets s_1, s_2 sont voisins dans un graphe G on peut écrire la fonction Python suivante :

```
def sontVoisins(s1,s2):  
    return s2 in listeVoisins(s1)
```

Ecrire une fonction Python `triangle(s1,s2,s3)` qui retourne `True` si les trois sommets s_1, s_2, s_3 forment un triangle, et `False` sinon :

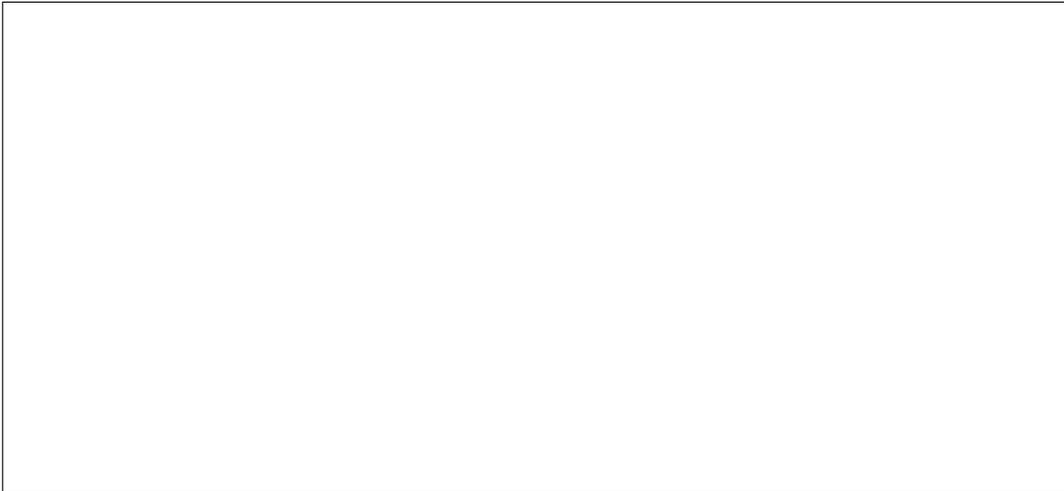
Exercice 3 On rappelle qu'un graphe G est *cubique* si tous les sommets de G sont de degré 3.

1. Dessiner un graphe cubique *simple* G_1 avec six sommets :

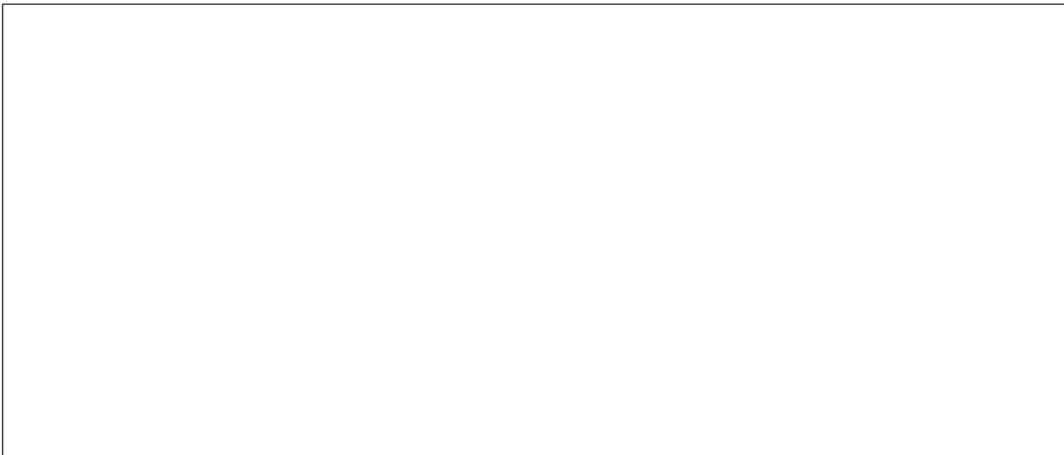
2. Prouver que tout graphe cubique *simple* avec six sommets est connexe :



3. Dessiner un graphe cubique *non connexe* G_2 avec six sommets :



4. Expliquer pourquoi on ne peut pas construire de graphe cubique avec cinq sommets :



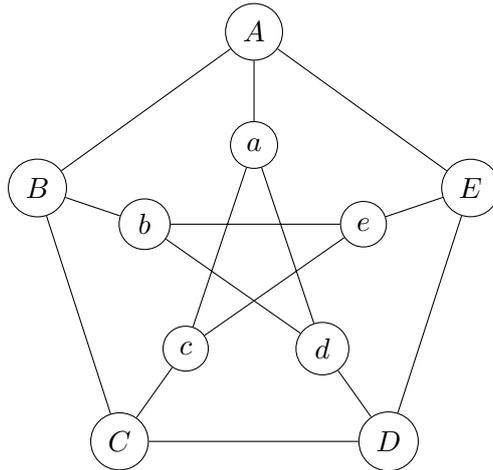
Exercice 4

1. Ecrire une fonction Python `existeSommetDegre(G,d)` qui retourne `True` s'il existe un sommet de degré d dans le graphe G , et `False` sinon :

2. Ecrire une fonction Python `nbSommetsDegre(G,d)` qui retourne le nombre de sommets de degré d dans le graphe G :

3. Ecrire une fonction Python `sommetVoisinDegre(s,d)` qui retourne un voisin de degré d du sommet s s'il en existe un, et `None` sinon :

Exercice 5 Voici le graphe de Petersen :



Dans cet exercice un *cycle* désigne toujours un cycle simple, c'est-à-dire un cycle dont tous les sommets sont distincts (sauf le dernier, qui est égal au premier).

1. $A - B - C - D - E - A$ est un cycle de longueur 5 (dans un graphe simple il n'est pas nécessaire d'indiquer l'arête qui relie deux sommets consécutifs du cycle). Donner deux autres cycles de longueur 5 :

2. Pour chaque entier k compris entre 6 et 10 indiquer si le graphe de Petersen possède un cycle de longueur k , et si oui donner un exemple :

| | |
|----------|--|
| $k = 6$ | |
| $k = 7$ | |
| $k = 8$ | |
| $k = 9$ | |
| $k = 10$ | |

Annexe : voici un rappel des principales fonctions disponibles pour manipuler les graphes (à vous de voir celles qui sont utiles pour ce devoir).

| | |
|-------------------------------------|--|
| L'argument G est un graphe | |
| <code>listeSommets(G)</code> | retourne la <i>liste</i> des <i>sommets</i> de G . |
| <code>nbSommets(G)</code> | retourne le <i>nombre</i> de <i>sommets</i> de G . |
| <code>sommetNom(G,etiquette)</code> | retourne le <i>sommet</i> de G désigné par son <i>nom</i> (étiquette). Exemple : <code>sommetNom (Europe, 'Italie')</code> . |
| <code>sommetNumero(G,i)</code> | retourne le <i>sommet</i> numéro i dans G ; la numérotation commence à 0. |

| | |
|----------------------------------|--|
| L'argument s est un sommet | |
| <code>listeVoisins(s)</code> | retourne la <i>liste</i> des <i>voisins</i> de s . |
| <code>degre(s)</code> | retourne le <i>degré</i> de s . |
| <code>nomSommet(s)</code> | retourne le <i>nom</i> (étiquette) de s . |
| <code>colorierSommet(s,c)</code> | colorie s avec la couleur c . Exemples de couleurs : 'red', 'green', 'blue', None. |
| <code>couleurSommet(s)</code> | retourne la <i>couleur</i> de s . |
| <code>marquerSommet(s)</code> | |
| <code>demarquerSommet(s)</code> | marque ou démarque s . |
| <code>estMarqueSommet(s)</code> | retourne True si s est marqué, False sinon. |