Université Bordeaux 1 — Devoir surveillé du 17 janvier 2013
Département Licence, **J1MI1003** — 11h-12h30

Numéro d'anonymat : Groupe :

Le sujet comporte 4 exercices et 6 pages, dont une page d'annexe.
Aucun document n'est autorisé : toutes les fonctions de manipulation des graphes disponibles sont rappelées en annexe, page 6, que vous pouvez détacher pour plus de facilités.

Exercice 1 Soit la fonction `Python` suivante :

```
def mystere(v):
    r = 0
    while v > 0 :
        if v % 10 == 0 :
            r = r + 1
        v = v // 10
    return r
```

Rappel : en python, le quotient entier de `a` par `b` est noté `a//b`, et le reste de la division entière est noté `a%b` (modulo).

1. Donner les valeurs successives prises par les variables `v` et `r` lors de l'appel `mystere(500601)`, ainsi que la valeur retournée.

v	500601	500601	50060	50060	5006	5006	500	500	50	50	5	5	0
r	0	0	0	1	1	1	1	2	2	3	3	3	3

3

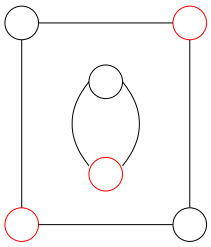
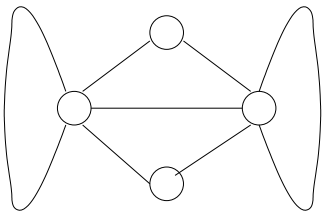
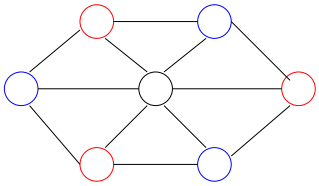
2. D'une manière générale, indiquer ce que retourne `mystere(v)` en fonction de l'entier strictement positif `v`.

`mystere(v)` retourne le nombre de chiffres zéro dans l'écriture décimale de l'entier `v`.

3. En vous inspirant de la fonction `mystere`, écrire une fonction `plusGrandChiffre(v)` qui retourne le plus grand chiffre contenu dans le nombre entier `v`. Par exemple, la valeur retournée par `plusGrandChiffre(500601)` est 6.

```
def plusGrandChiffre(v):
  max = 0
  while v > 0:
    if v % 10 > max:
      max = v % 10
    v = v // 10
  return max
```

Exercice 2 On rappelle que K_6 désigne le graphe complet à 6 sommets. On appelle F, G et H les graphes suivants :

F =  G =  H = 

1. Remplir le tableau suivant (en répondant par OUI ou par NON lorsque cela est demandé). **Attention**, les réponses fausses enlèvent des points :

Graphe	F	G	H	K_6
Connexe ? Si non, justifier	NON	OUI	OUI	OUI
Simple ? Si non, justifier	NON, arête multiple au milieu	NON, boucles à droite et à gauche	OUI	OUI
Eulérien ? (justifier)	NON non connexe	OUI connexe 2 sommets impairs	NON 6 sommets impairs	NON 6 sommets impairs
Peut-il être bien colorié ? Si non, justifier	OUI	NON, présence de boucles	OUI	OUI
Si oui, nombre chromatique ? (justifier)	2 : voir dessin		3 : contient K_3 et voir dessin	6 : est K_6

2. Un graphe composé seulement d'un cycle est-il toujours 2-coloriable ? Justifier.

Non. Par exemple, K_3 un composé seulement d'un cycle, mais n'est pas 3-coloriable. Il faut et il suffit que le cycle soit de longueur paire.

Numéro d'anonymat : Groupe :

Exercice 3

1. Écrire une fonction `compterAretesMarquees(G)` qui renvoie le nombre d'arêtes marquées du graphe `G`.

```
def compterAretesMarquees(G):
  n = 0
  for s in listeSommets(G):
    for a in listeAretesIncidentes(s):
      if estMarqueeArete(a):
        n = n + 1
  return n // 2
```

2. Écrire une fonction `marquerAretesBicolores(G)` qui marque les arêtes du graphe `G` pour lesquelles les deux sommets incidents sont coloriés avec une couleur distincte. On supposera que `'white'` est aussi une couleur.

```
def marquerAretesBicolores(G):
  for s in listeSommets(G):
    c = couleurSommet(s)
    for a in listeAretesIncidentes(s):
      s2 = sommetVoisin(s,a)
      if couleurSommet(s2) != c:
        marquerArete(a)
```

3. Écrire une fonction `sommetEstNonMarqueAvecAreteMarquee(s)` qui renvoie `True` si le sommet `s` est non marqué et qu'il est incident à au moins une arête marquée, et renvoie `False` sinon.

```
def sommetEstNonMarqueAvecAreteMarquee(s):
  if estMarqueSommet(s):
    return False
  for a in listeAretesIncidentes(s):
    if estMarqueeArete(a):
      return True
  return False
```

4. Écrire une fonction `listeSommetsNonMarquesAvecAretesMarquees(G)` qui renvoie la liste des sommets du graphe `G` qui sont non marqués et incidents à au moins une arête marquée. La fonction précédente pourra être utilisée, même si vous n'avez pas su l'écrire.

```
def listeSommetsNonMarquesAvecAretesMarquees(G):
  L = []
  for s in listeSommets(G):
    if sommetEstNonMarqueAvecAreteMarquee(s):
      L = L + [ s ]
  return L
```

5. Écrire une fonction `marquerSommetsAvecAretesMarquees(G)` qui marque tous les sommets du graphe `G` dont au moins une arête incidente est marquéee. On pourra utiliser une fonction précédente, même si vous n'avez pas su l'écrire.

```
def marquerSommetsAvecAretesMarquees(G):
  for s in listeSommets(G):
    if sommetEstNonMarqueAvecAreteMarquee(s):
      marquerSommet(s)
```

ou

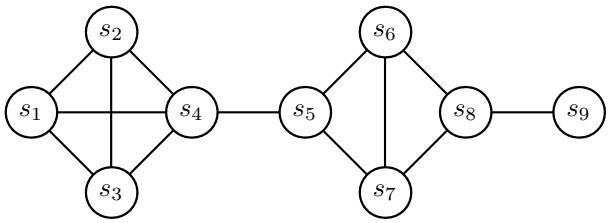
```
def marquerSommetsAvecAretesMarquees(G):
  L = listeSommetsNonMarquesAvecAretesMarquees(G)
  for s in L:
    marquerSommet(s)
```

Exercice 4 Une **clique** dans un graphe G est un sous-ensemble de sommets C deux-à-deux adjacents, i.e., tel que pour toute paire de sommets distincts de C, il existe une arête dans G reliant les deux sommets, i.e. c'est un sous-graphe complet.

Une clique C est *maximale* si pour tout sommet s qui n'est pas dans C, $C \cup \{s\}$ n'est plus une clique.

Voici le graphe H.

H



Numéro d'anonymat : Groupe :

1. Est-ce que $\{s_1, s_2, s_3\}$ est une clique de H ? Une clique maximale ? Sinon, trouver dans H une clique maximale contenant $\{s_1, s_2, s_3\}$.

   $\{s_1, s_2, s_3\}$ est une clique, mais elle n'est pas maximale. $\{s_1, s_2, s_3, s_4\}$ est une clique maximale contenant $\{s_1, s_2, s_3\}$

2. Déterminer toutes les cliques maximales du graphe H (il y en a 5).

   $\{s_1, s_2, s_3, s_4\}$, $\{s_4, s_5\}$, $\{s_5, s_6, s_7\}$, $\{s_6, s_7, s_8\}$, $\{s_8, s_9\}$

3. Donner trois cliques disjointes (c'est-à-dire sans aucun sommet commun) qui ensemble couvrent tous les sommets de H.

   $\{s_1, s_2, s_3, s_4\}$, $\{s_5, s_6, s_7\}$, $\{s_8, s_9\}$

4. Est-ce qu'un graphe simple peut contenir une clique maximale à un seul sommet ? Si c'est le cas, quel est le degré d'un tel sommet ?

   Oui : le graphe réduit à un seul sommet sans arête, par exemple. Il est alors de degré 0.
   S'il avait une arête incidente, en ajoutant le sommet voisin, on obtiendrait une clique strictement plus grande, notre clique ne serait donc pas maximale. Il ne peut donc pas y avoir d'arête incidente, le sommet est forcément de degré 0.

5. Est-il possible qu'un graphe simple ne possède qu'une seule clique maximale ? Si c'est le cas, de quel(s) graphe(s) s'agit-il ?

   Oui : un graphe complet. L'ensemble de tous ses sommets est une clique. Elle contient déjà tous les sommets, donc ne peux pas être agrandie, et elle contient toutes les cliques, donc c'est la seule pouvant être maximale.
   Lorsqu'un graphe ne possède qu'une clique maximale, celle-ci contient forcément tous les sommets : s'il existait un sommet en-dehors de cette clique maximale, on pourrait construire une autre clique maximale le contenant.
   Un graphe simple avec une seule clique maximale est donc forcément un graphe complet K_n.

Fin.

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	
`listeSommets(G)`	retourne la *liste* des *sommets* de G.
`nbSommets(G)`	retourne le *nombre* de *sommets* de G.
`sommetNom(G,etiquette)`	retourne le *sommet* de G désigné par son *nom* (étiquette). Exemple : `sommetNom (Europe, 'Italie').`
`sommetNumero(G,i)`	retourne le *sommet* numéro i dans G ; la numérotation commence à 0.

L'argument s est un sommet	
`listeVoisins(s)`	retourne la *liste* des *voisins* de s.
`degre(s)`	retourne le *degré* de s.
`nomSommet(s)`	retourne le *nom* (étiquette) de s.
`colorierSommet(s,c)`	colorie s avec la couleur c. Exemples de couleurs : `'red'`, `'green'`, `'blue'`, `None`.
`couleurSommet(s)`	retourne la *couleur* de s.
`marquerSommet(s)` `demarquerSommet(s)`	marque ou démarque s.
`estMarqueSommet(s)`	retourne `True` si s est marqué, `False` sinon.
`listeAretesIncidentes(s)`	retourne la *liste* des arêtes *incidentes* à s.

L'argument a est une arête	
`nomArete(a)`	retourne le *nom* (étiquette) de a.
`marquerArete(a)` `demarquerArete(a)`	marque ou démarque a.
`estMarqueeArete(a)`	retourne `True` si a est marquée, `False` sinon.

Arguments : un sommet s et une arête a	
`sommetVoisin(s,a)`	retourne le voisin de s en suivant l'arête a.