Numéro d'anonymat :

Université Bordeaux 1 — Devoir Surveillé Terminal du 16 janvier 2014
Département Licence, **J1MI1003** — 11h-12h30

Le sujet comporte 7 exercices et 7 pages, dont une page d'annexe

Aucun document n'est autorisé : toutes les fonctions de manipulation des graphes disponibles sont rappelées en annexe page 7. Vous pouvez détacher cette annexe pour plus de facilité.

Exercice 1 On considère la fonction suivante :

```
def mystere (a, b):
   while b > 0:
      p = a % b
      a = b
      b = p
   return a
```

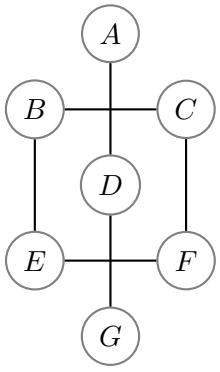
1. Simuler l'exécution de `mystere(48,21)` en complétant le tableau suivant :

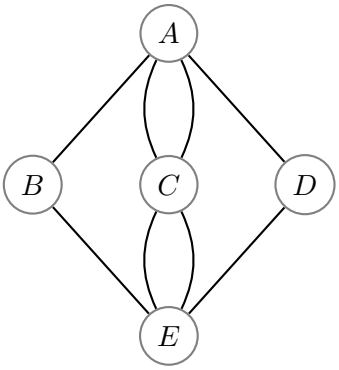
a	
b	
p	

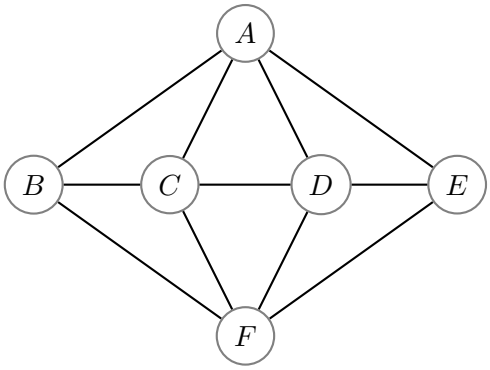
2. Que vaut `mystere(48,21)` ? ☐

3. D'une manière générale, indiquer ce que retourne `mystere(a,b)` en fonction des entiers naturels a et b.

Exercice 2 On note respectivement G_0, G_1 et G_2 les trois graphes suivants :



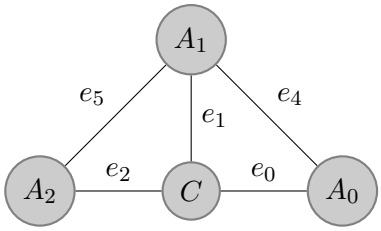




Remplir le tableau suivant (en répondant par OUI ou par NON lorsque cela est demandé). **Attention**, les réponses fausses enlèvent des points.

Graphe	G_0	G_1	G_2
Connexe ? Si non, justifier			
Simple ? Si non, justifier			
Eulérien ? (justifier)			
nombre chromatique ? (justifier)			

Exercice 3 Dans un graphe, un *triangle* est un cycle élémentaire de longueur 3. Par exemple, dans le graphe T suivant :



le cycle $[C, e_0, A_0, e_4, A_1, e_1, C]$ est un triangle.

1. Écrire une fonction `triangle(x,y,z)` qui prend en arguments trois sommets d'un graphe, et retourne `True` si le graphe comporte un triangle passant par ces trois sommets, et `False` sinon.
   Indication : on pourra utiliser, sans la réécrire, la fonction `sontVoisins(s,t)` vue en cours, qui prend en arguments deux sommets d'un graphe, et retourne `True` si ces sommets sont voisins, et `False` sinon.

2. Le graphe T contient 2 triangles différents : indiquer lesquels.

3. Écrire une fonction `nombreTriangles(G)` prenant en argument un graphe simple G, et retournant le nombre de triangles contenus dans G.

Exercice 4 Indiquer si les propositions suivantes sont vraies ou fausses en justifiant votre réponse.

— Soit G un graphe. Si tout sommet de G est contenu dans un cycle élémentaire de longueur non nulle, alors G est connexe.

— Soit G un graphe. Si pour chaque paire de sommets u, v de G il existe un cycle élémentaire de longueur non nulle les contenant, alors G est connexe.

— Soit G un graphe à au moins deux sommets. Si G ne contient aucun cycle élémentaire de longueur non nulle, alors G contient (au moins) un sommet de degré 1.

— Soit G un graphe. Si tout sommet de G est contenu dans un cycle élémentaire de longueur non nulle, alors G ne contient aucun sommet de degré 1.

Exercice 5 Une *étoile* à n sommets est un graphe simple, ayant un sommet de degré $n-1$ et $n-1$ sommets de degré 1.

1. Dessiner une *étoile* à 5 sommets.

2. En utilisant les fonctions Python nécessaires de l'annexe et la fonction `estSimple(G)` écrite en TD (qui retourne `True` si le graphe `G` est simple, `False` sinon), écrire une fonction `estEtoile(G)` qui prend en argument un graphe, et retourne `True` si le graphe est une *étoile*, et `False` sinon.

3. Démontrer que toute *étoile* à n sommets est un graphe connexe.

4. Montrer que toute *étoile* à n sommets est un *arbre* à n sommets.

Exercice 6 Écrire une fonction `monoCouleur(G)` qui prend en argument un graphe, et retourne `True` si les sommets du graphe sont tous de la même couleur, et `False` sinon.

Exercice 7 On cherche à caractériser les arbres eulériens.

1. Montrer que si un arbre admet une chaîne eulérienne, cette chaîne passe une fois et une seule par chaque sommet de l'arbre.

2. En déduire la forme générale des arbres eulériens.

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* (`etiquette`). Exemple : `sommetNom(Europe, 'Italie')`
`dessinerGraphe(G)` ou simplement `dessiner(G)`	demande (très poliment) au logiciel *Graphviz* de dessiner le graphe `G` ; voir page suivante pour les détails

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'`, `'white'`, `'cyan'`, `'yellow'`
`couleurSommet(s)`	retourne la *couleur* de `s`
`marquerSommet(s)` `demarquerSommet(s)`	marque le sommet `s` démarque le sommet `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 l'arête `a` démarque l'arête `a`
`estMarqueeArete(a)`	retourne `True` si `a` est marquée, `False` sinon

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

L'argument `a` est une liste	
`elementAleatoireListe(u)`	retourne un élément choisi aléatoirement dans la liste `u` si celle-ci est non vide. Si la liste `u` est vide la fonction retourne une erreur `IndexError`. Exemple : `elementAleatoireListe(listeSommets(tgv2005))`