Université Bordeaux 1 — Devoir surveillé du 19 novembre 2012
Département Licence, **J1MI1003** — 14h-15h30

Nom : Prénom : Groupe :

Le sujet comporte 4 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 cet annexe pour plus de facilité.

Exercice 1 Soit un graphe à 6 sommets, dont voici les listes de voisins :

- voisins de A : [C]
- voisins de B : [C, C, F]
- voisins de C : [A, B, B, F]
- voisins de D : [D, D]
- voisins de E : [E, E, F]
- voisins de F : [B, C, E]

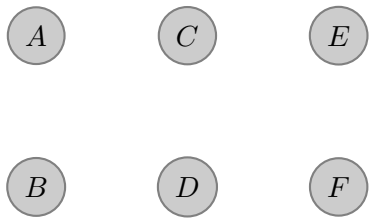
1. Donner le degré de chaque sommet du graphe :

d(A)= , d(B)= , d(C)= , d(D)= , d(E)= , d(F)=

2. Le graphe possède-t-il des boucles ? Si oui combien et pour quel(s) sommet(s) ?

3. Le graphe possède-t-il des arêtes multiples ? Si oui combien et pour quels sommets ?

4. Compléter le dessin de ce graphe.



5. Après avoir numéroté les différentes arêtes du graphe, donner une chaîne élémentaire de longueur maximale.

Exercice 2 Voici une fonction Python :

```
def mystere(n, k):
    p = 1
    i = 0
    while i < k:
        p = p * (n - i)
        p = p / (i + 1)
        i = i + 1
    return p
```

1. Simuler l'exécution de `mystere(6,3)` en donnant les valeurs successives des variables `p` et `i` dans le tableau suivant :

   $n = 6, k = 3$

p	
i	

2. D'une manière générale, indiquer ce que retourne `mystere(n,1)` en fonction de l'entier `n` (on suppose `n` > 0). Même question pour `mystere(n,2)`.

3. D'une manière générale, indiquer ce que retourne `mystere(n,k)` en fonction des entiers strictement positifs `n` et `k`.

4. Réécrire la fonction mystère en utilisant une boucle `for` plutôt qu'une boucle `while`.

Nom : Prénom : Groupe :

Exercice 3

1. Écrire une fonction `couleurDifferenteDesVoisins(s)` qui renvoie `True` si le sommet `s` est colorié avec une couleur distincte de celle de chacun de ses voisins, et renvoie `False` sinon. Dans tout cet exercice on supposera que `'white'` est aussi une couleur.

Un graphe est dit bien colorié si deux sommets voisins ont toujours des couleurs différentes.

2. En réutilisant la fonction précédente, écrire une fonction `bienColorie(G)` qui teste si un graphe `G` est bien colorié.

3. Écrire une fonction `nbSommetsMalColories(G)` qui renvoie le nombre de sommets mal coloriés du graphe `G` (un sommet est mal colorié s'il a la même couleur qu'un de ses voisins).

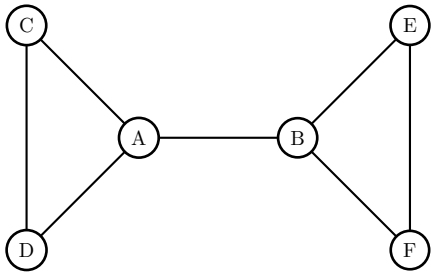
4. Écrire une fonction `nbAretesBicolores(G)` qui renvoie le nombre d'arêtes bicolores du graphe `G` (une arête est bicolore si ses extrémités sont de couleurs différentes).

Exercice 4 Soit G un graphe simple connexe.

Dans G, un isthme est une arête dont la suppression détruit la connexité du graphe.

Dans G, un point d'articulation est un sommet dont la suppression détruit la connexité du graphe (la suppression d'un sommet s enlève également les arêtes incidentes à s).

Par exemple, le graphe suivant a **1 isthme** et **2 points d'articulation** :



L'arête A–B est un isthme (si on l'enlève, les sommets A, C et D sont déconnectés des sommets B, E et F) ; c'est le seul (si on enlève n'importe quelle autre arête, le graphe reste connexe). Les points d'articulation sont A et B.

1. Dessiner un graphe simple connexe admettant exactement 2 isthmes et 1 point d'articulation (les mettre en évidence).

Nom : Prénom : Groupe :

2. Dessiner un graphe simple connexe admettant un point d'articulation (le mettre en évidence), mais n'admettant aucun isthme.

3. Peut-on dessiner un graphe simple connexe admettant un isthme, mais n'ayant aucun point d'articulation ? Justifier votre réponse par un exemple ou une explication.

4. Un graphe simple connexe dont tous les sommets sont de degré pair peut-il avoir un point d'articulation ? Justifier votre réponse par un exemple ou une explication.

5. Un graphe simple connexe dont tous les sommets sont de degré pair peut-il avoir un isthme ? Justifier votre réponse par un exemple ou une explication.

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`