Université Bordeaux 1 — Devoir surveillé du 18 novembre 2013
Département Licence, **J1MI1003** — 14h-15h30

Nom : Prénom : Groupe :

Le sujet comporte 6 exercices et 5 pages, dont une page d'annexe

Aucun document n'est autorisé : toutes les fonctions de manipulation des graphes disponibles sont rappelées en annexe page 5. Vous pouvez détacher cette annexe pour plus de facilité.

Exercice 1 Soit un graphe à 8 sommets, dont voici les listes de voisins :

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

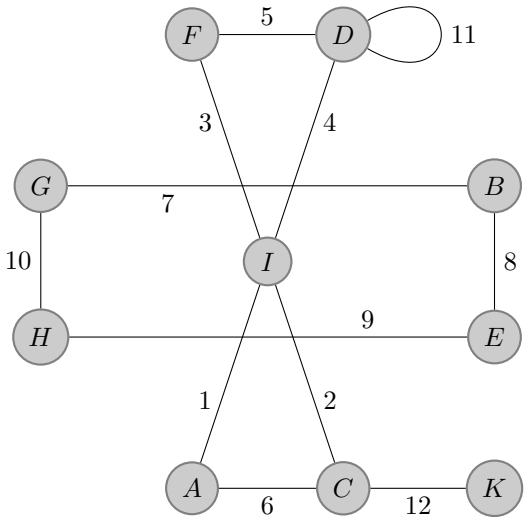
Le graphe possède-t-il des boucles ? Si oui combien et pour quel(s) sommet(s) ?

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

Combien le graphe possède-t-il de composantes connexes ? Listez-les.

Dessiner ce graphe.

Exercice 2 Soit le graphe ainsi représenté :



1. Donner une chaîne élémentaire de longueur maximale.

2. Donner un cycle élémentaire de longueur maximale.

Exercice 3 Pour tester si deux sommets s_1 et s_2 sont voisins dans un graphe G, on peut écrire la fonction Python suivante, que l'on pourra appeler pour répondre à cet exercice :

```
def sontVoisins(s1, s2):
  for s in listeVoisins(s1):
    if s == s2:
      return True
  return False
```

Écrire une fonction `estVoisinDeTous(s,G)` qui teste si le sommet `s` est voisin avec tous les autres sommets de `G`.

Exercice 4 (Chaque question de cet exercice peut-être traitée de manière indépendante des autres : vous pouvez utiliser les fonctions des questions précédentes, même si vous n'avez pas réussi à les implémenter).

1. Écrire une fonction `sommeDiviseurs(n)` qui retourne la somme des diviseurs de `n`.

2. On dit qu'un nombre est *parfait* s'il est égal à la somme de ses diviseurs (strictement) plus petits que lui. Par exemple, 6 est un nombre parfait car 6=1+2+3. Les nombres 15 et 28 sont-ils parfaits ? Justifiez vos réponses.

3. Ecrire une fonction `estParfait(n)` qui retourne `True` si `n` est un nombre parfait, `False` sinon.

4. Ecrire une fonction `prochainParfait(n)` qui retourne le plus petit nombre parfait supérieur ou égal à `n`.

Exercice 5 Soit `G` un graphe et `c` une couleur, écrire une fonction `listeSommetsDeCouleur(G,c)` qui renvoie la liste des sommets de `G` colorés avec la couleur `c`. Si aucun sommet n'est de couleur `c`, la fonction renverra la liste vide. On considère ici que `white (blanc)` est aussi une couleur et sera traité comme les autres couleurs.

Exercice 6 Voici une fonction `Python` :

```
def f(n,k):
    c = 1
    for i in range(1,k+1):
        c = c * (n - 1 - i) // i
    return c
```

1. Simuler l'exécution de `f(7,3)` en complétant le tableau suivant :

i	1
c	1

2. Que vaut `f(7,3)` ?

3. Écrire une fonction Python `g(n)` qui retourne $\sum_{k=0}^{k=n}$ `f(n,k)` :

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`