Numéro d'anonymat :

Université de Bordeaux
Département Licence, **4TPM101U**

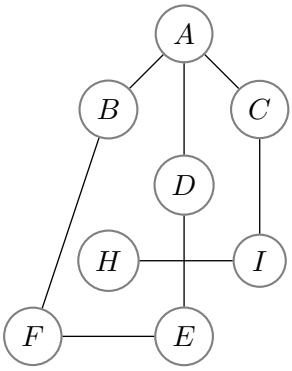
Examen du 19 janvier 2018
11h30-13h

Le sujet comporte 7 exercices et 9 pages, dont une page d'annexe

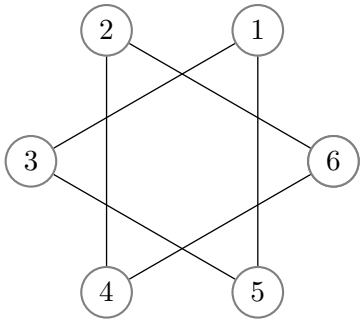
Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images et de graphes disponibles sont rappelées en annexe page 9. Vous pouvez détacher cette annexe pour plus de facilité.

Exercice 1 Les questions de cet exercice portent toutes sur ces trois graphes : G_0, G_1 et G_2.

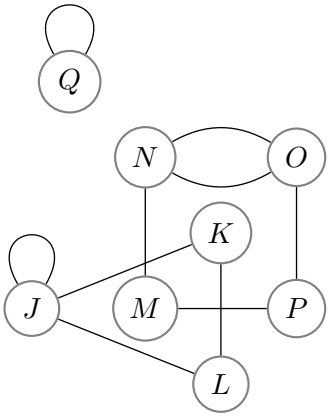
Graphe G_0



Graphe G_1



Graphe G_2



1. Remplir le tableau ci-dessous.

	G_0	G_1	G_2
Donner les ensembles de sommets correspondant aux composantes connexes.			
Donner un sommet de degré maximal et préciser son degré.			
Donner une chaîne élémentaire de longueur maximale.			

Exercice 2 On considère la fonction suivante :

```
def mystere(n):
    k = n
    while k%2 == 0:
        k = k//2
    return k == 1
```

1. Simuler l'exécution de `mystere(240)` en complétant le tableau suivant :

k		

   Que vaut `mystere(240)` ?

2. Que vaut `mystere(256)` ?

3. Expliquer ce que retourne `mystere(n)` en fonction de l'entier naturel n.

Exercice 3 Somme des lancés d'un dé
On considère un dé à 6 faces simulé par un appel à
`elementAleatoireListe(list(range(1,7)))`.
Écrire une fonction `sommeSuperieureA(n)` qui retourne le nombre de lancés effectués pour que la somme des chiffres obtenus soit strictement supérieure à l'entier `n` passé en paramètre.

Exercice 4 Ensemble de sommets dominant d'un graphe

1. Écrire une fonction `sommetsTousMarques(G)` qui teste si tous les sommets du graphe `G` sont marqués.

2. Écrire une fonction `marquerSommets(G, U)` qui prend en paramètre un graphe `G` et une liste `U` de sommets de ce graphe, et qui, après avoir démarqué tous les sommets de `G`, marque tous les sommets de la liste `U` et tous les voisins des sommets de la liste `U`.

3. Dans un graphe `G`, un ensemble de sommets `U` est un ensemble dominant de `G` si pour chaque sommet `s` du graphe, `s` fait partie de l'ensemble `U` ou `s` est voisin d'un sommet faisant partie de `U`.

   Soient les ensembles $E_1 = \{A, B, C, D, E, F, H, I\}$, $E_2 = \{A, C, E, I\}$, $E_3 = \{A, E, H, I\}$, $E_4 = \{B, C, D, E\}$, $E_5 = \{D, F, I\}$, et $E_6 = \{A, D, I\}$. Lister les ensembles E_i ne dominant pas le graphe G_0 de l'exercice 1. Expliquer pourquoi ils ne sont pas dominants.

4. Écrire une fonction `dominant(G,U)` qui teste si `U` (que l'on supposera représenté par une liste) est un ensemble dominant de `G` en utilisant l'algorithme qui consiste à vérifier qu'après avoir marqué tous les sommets de `U` et tous leurs voisins, tous les sommets du graphe sont alors marqués.

Exercice 5 Dynamique d'une image en niveau de gris

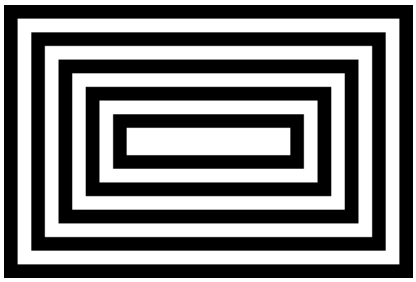
On dispose d'une image en niveau de gris, c'est-à-dire que tous les pixels ont une couleur RGB de la forme (ng, ng, ng). La dynamique d d'une telle image est définie comme la différence entre le niveau de gris maximum des pixels de l'image ng_{max} et le niveau de gris minimum des pixels de l'image ng_{min}. On a donc $d = ng_{max} - ng_{min}$. Cette mesure permet d'évaluer le contraste d'une image.

Écrire une fonction `dynamique(img)` qui retourne la dynamique de l'image `img`. Votre algorithme ne devra utiliser qu'un seul parcours de l'image : le niveau de gris d'un pixel donné sera lu une seule fois.

Exercice 6

1. Ecrivez une fonction `rectanglePlein(img,x1,x2,y1,y2,c)` qui dessine un rectangle plein de couleur `c` dont les coins sont les pixels de coordonnées $(x1, y1)$, $(x2, y1)$, $(x1, y2)$, $(x2, y2)$.

2. Ecrivez une fonction `rectanglesEmpiles(img, d)` qui dessine dans l'image `img` une suite de rectangles pleins noirs et blancs de façon à obtenir l'effet suivant :



On se sert du fait qu'un nouveau dessin cache le dessin précédent. Si un rectangle est de largeur ℓ pixels et de hauteur h pixels alors le rectangle dessiné après lui sera de largeur $\ell - 2d$ pixels et de hauteur $h - 2d$ pixels (on réduit la taille d'une bande de d pixels sur les quatre côtés).

Exercice 7 Nombre d'arêtes d'un graphe sans cycle élémentaire

Étant donné un graphe G et une paire de sommets u et v, on notera $G+uv$ le graphe obtenu à partir de G en ajoutant une nouvelle arête reliant les sommets u et v.

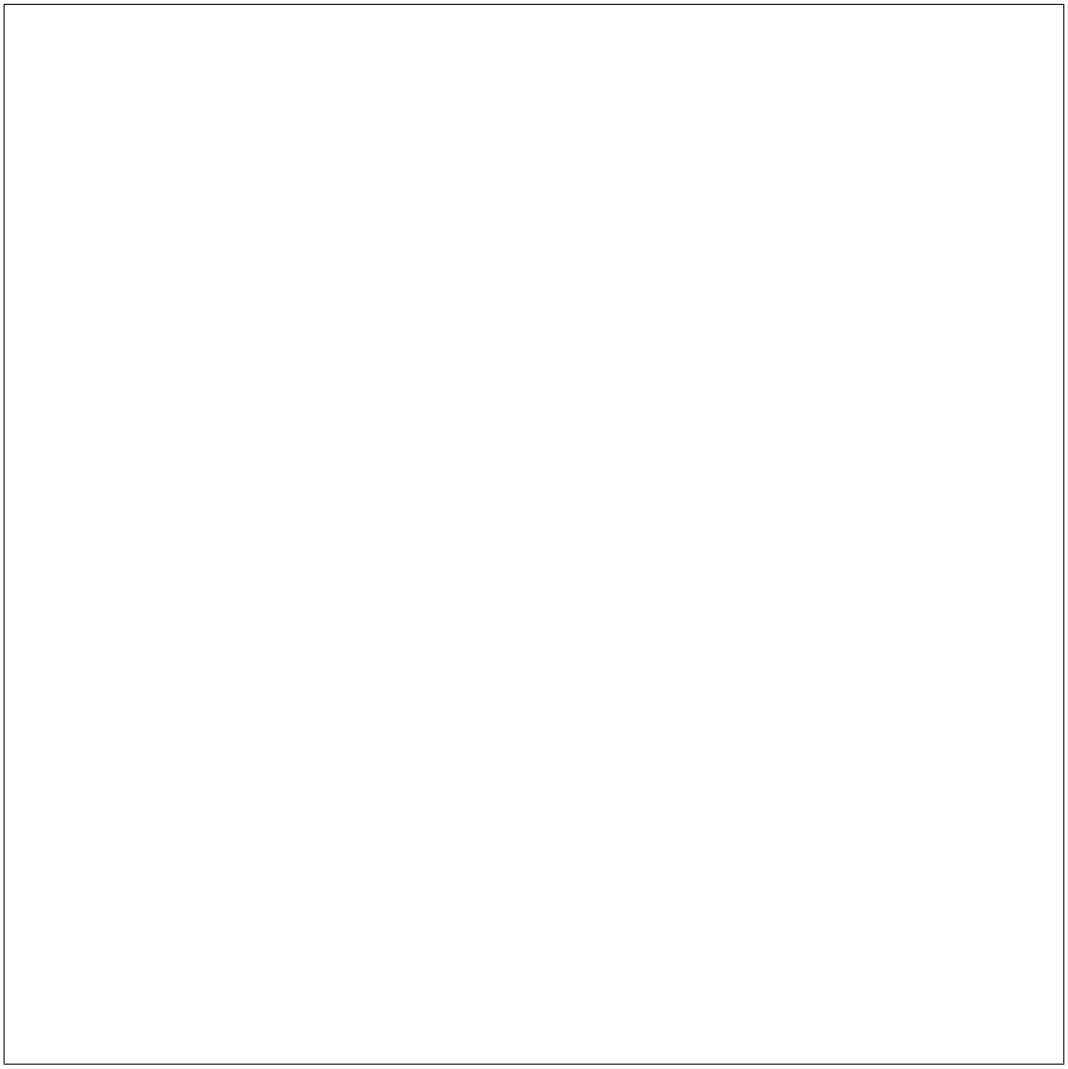
1. Pour chacune des propositions suivantes justifier brièvement pourquoi elle est vraie :

   (a) Soit G un graphe pas forcément connexe. Si deux sommets u et v de G appartiennent à la même composante connexe de G, alors le graphe $G + uv$ possède autant de composantes connexes que G.

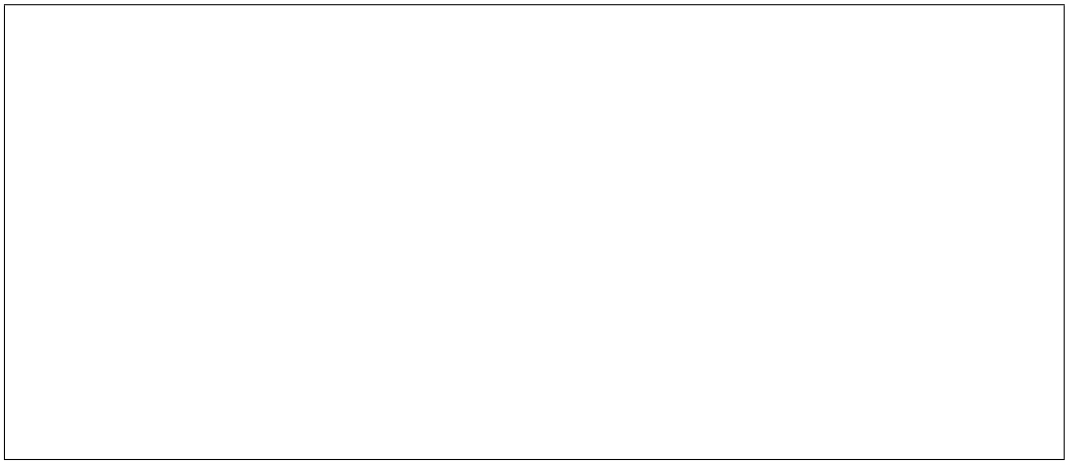
   (b) Soit G un graphe non connexe. Si deux sommets u et v de G appartiennent à deux composantes connexes différentes de G, alors le graphe $G+uv$ possède une composante connexe de moins par rapport à G.

   (c) Soit G un graphe connexe. Si u et v sont deux sommets de G, alors le graphe $G + uv$ contient au moins un cycle élémentaire passant par u et v.

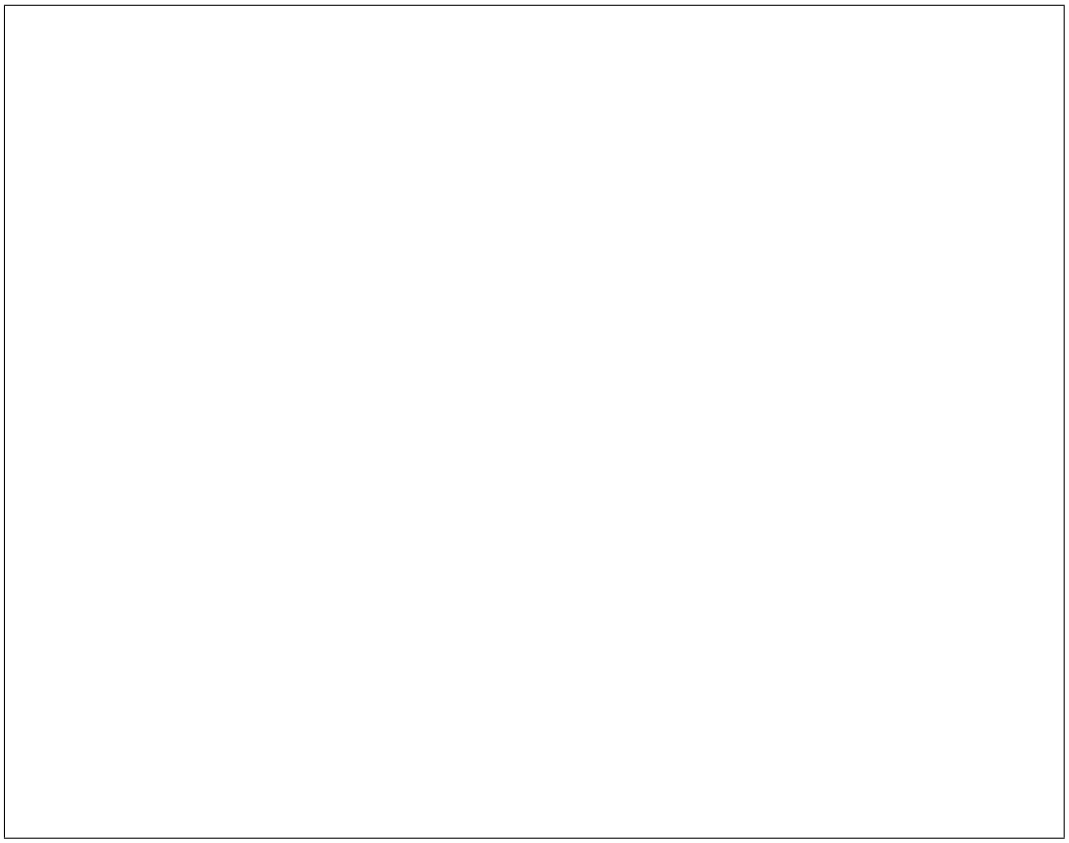
   (d) Soit G_0 un graphe sans arête (un graphe dont tous les sommets sont de degré 0). Alors G_0 a autant de composantes connexes que de sommets.



2. Soit G un graphe non connexe ne contenant aucun cycle élémentaire. Soient u et v deux sommets de G appartenant à deux composantes connexes différentes. Montrer avec un raisonnement par l'absurde que le graphe $G + uv$ ne contient pas de cycle élémentaire non plus.



3. Soit G un graphe connexe à n sommets ne contenant aucun cycle élémentaire de longueur non nulle. Soit G_0 un graphe sur le même ensemble de sommets que G, ne contenant aucune arête. On reconstruit G à partir de G_0 en ajoutant les arêtes de G une par une. En utilisant les résultats précédents, montrer qu'à chaque étape (après tout ajout d'une arête) le nombre de composantes connexes diminue exactement de 1. En déduire que le nombre d'arêtes de G est égal au nombre de sommets de G moins un.



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

L'argument `img` est une image	
`open(nom)`	Ouvre le fichier *nom* et retourne l'image contenue dedans (par exemple `open("teapot.png")`.
`Image.save(img, nom)`	Sauvegarde l'image *img* dans le fichier *nom*.
`new("RGB",(largeur,hauteur))`	Retourne une image de taille $largeur \times hauteur$, initialement noire.
`largeur = img.width` `hauteur = img.height`	Récupère la largeur de *img*. Récupère la hauteur de *img*.
`Image.putpixel(img, (x,y), (r,g,b))`	Peint le pixel (x, y) dans l'image *img* de la couleur (r, g, b)
`(r,g,b) = Image.getpixel(img, (x,y))`	Retourne la couleur du pixel (x, y) dans l'image *img*

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')`

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`