Numéro d'anonymat :

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

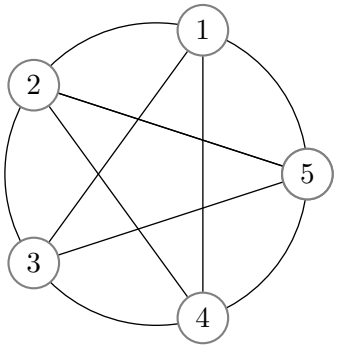
Examen du 23 janvier 2020
14h30-16h00

Le sujet comporte 5 exercices et 9 pages, dont une page d'annexe

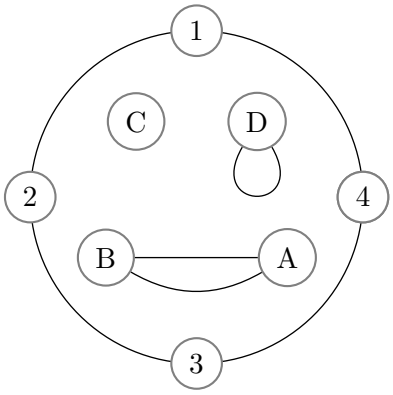
Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images et de graphes disponibles sont rappelées en annexe page 9. Vous pouvez détacher cette annexe pour plus de facilité.

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

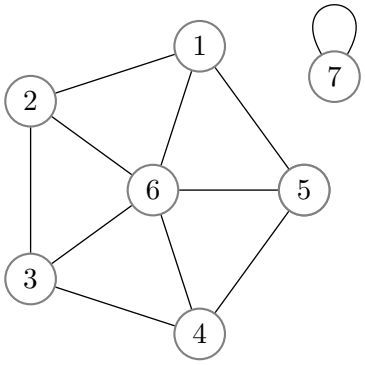
Graphe G_0



Graphe G_1



Graphe G_2



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 un cycle élémentaire de longueur maximale.			

Exercice 2 On considère la fonction suivante prenant comme paramètres un entier naturel n et un chiffe d (on a donc $0 \leq d \leq 9$).

```
def mystere(n,d):
    k = 0
    p = 0
    while n > 0 :
        ch = n%10
        k = k + 1
        if  ch == d:
            p = k
        n = n // 10
    return p
```

1. Simuler l'exécution de `mystere(3767,7)` en complétant le tableau suivant (pour simplifier, n'écrire que les changements de variable) :

n		
k		
p		
ch		

2. Que vaut `mystere(3767,7)` ?
3. Que vaut `mystere(5236,7)` ?
4. D'une manière générale, indiquer ce que retourne `mystere(n)` en fonction de l'entier naturel n et du chiffre d.

Exercice 3 L'objectif de cet exercice est d'écrire une fonction permettant d'incruster une image `img1` en filigrane dans une autre image `img2`, par exemple les images `logo` et `route` de la figure 1. On supposera que la largeur et la hauteur de `img1` sont inférieures ou égales à celles de `img2`.

L'algorithme utilisé est le suivant : pour chaque pixel de coordonnées (x, y) dans `img1`, on remplace la couleur initiale du pixel de coordonnées (x, y) dans `img2` par la moyenne des couleurs de ces deux pixels — la moyenne entre deux couleurs (r_1, g_1, b_1) et (r_2, g_2, b_2) étant égale à $(\frac{r_1+r_2}{2}, \frac{g_1+g_2}{2}, \frac{b_1+b_2}{2})$. L'image `img1` apparaît ainsi dans le coin en haut à gauche de l'image `img2`.

(a) `logo`

(b) `route`

FIGURE 1 – Images de départ

1. Écrire la fonction `filigrane(img1, img2)` qui modifie `img2` *selon l'algorithme précédent.* Par exemple, l'appel à `filigrane(logo, route)` transformera l'image `route` ainsi :

2. On souhaite désormais que l'image `img1` soit incrustée en bas à droite de l'image `img2`. Écrire la fonction `filigraneBasDroite(img1, img2)` produisant un tel résultat. Par exemple, l'appel à `filigraneBasDroite(logo, route)` transformera l'image `route` ainsi :

Exercice 4 1. Un graphe est dit *régulier* si tous ses sommets sont de même degré. Écrire la fonction `regulier(G)` qui retourne `True` si le graphe passé en paramètre est régulier et `False` sinon. Indice : la fonction `elementAleatoireListe(L)` pourra être employée.

2. Un graphe est dit *bien colorié* si deux sommets voisins ont toujours des couleurs différentes. Écrire la fonction `bienColorie(G)` qui retourne `True` si le graphe passé en paramètre est bien colorié et `False` sinon.

Exercice 5 – Raisonnement sur les graphes

1. Dans une soirée, tous les invités serrent la main à trois personnes sauf une qui ne serre la main qu'à une seule personne (bien entendu on ne resserre pas plusieurs fois la main d'une même personne). Peut-il y avoir 21 personnes ?

   Oui ..... ☐ Non ..... ☐

   Justifier obligatoirement votre réponse :

2. Soit G un graphe avec n sommets et m arêtes, dont tous les sommets ont pour degré au moins δ. Montrer que n est inférieur ou égal à $2m/\delta$.

3. Soit G un graphe ayant $2p$ sommets, que l'on suppose simple, c'est-à-dire sans boucle ni arête multiple. Démontrer (par un raisonnement par l'absurde) que si le degré de chaque sommet est au moins égal à p, alors ce graphe est connexe.

4. On a construit des ponts (sur lesquels on peut circuler dans les deux sens) entre les îles d'un archipel de sorte de pouvoir aller (directement ou indirectement) de toute île à une autre. **Il peut y avoir plusieurs ponts entre deux mêmes îles**. De plus, de chaque île part un **nombre pair de ponts**. On a remarqué que, lorsqu'un pont est impraticable pour cause de travaux (**quel que soit ce pont**), **on peut encore aller de toute île à une autre**. Dans la suite on modélise les îles par des sommets d'un graphe et les ponts par des arêtes. Dans ce cas,

(a) dessiner ci-après en `(fig. a)` un archipel de trois îles qui utilise exactement 3 ponts ;

(b) dessiner ci-après en `(fig. b)` un archipel de trois îles qui utilise exactement 4 ponts.

`(fig. a)`	`(fig. b)`

On supposera que l'archipel est modélisé par un graphe. Dans ce cas, pour chaque propositions cochez la case `V` si vous pensez que la proposition est vraie, cochez la case `F` si vous pensez que la proposition est fausse et si vous ne savez pas, ne cochez ni `V` ni `F` (car une mauvaise réponse sera pénalisée).

`V` / `F`

☐ / ☐ Le graphe est simple, i.e. sans boucle ni arête multiple.

☐ / ☐ Il y a au plus une île d'où partent exactement 4 ponts.

☐ / ☐ Il y a au moins une île d'où partent exactement 2 ponts.

☐ / ☐ Le graphe est connexe et tout sommet est de degré pair, la suppression d'une arête ne détruit pas la connexité du graphe.

☐ / ☐ Le graphe est connexe et tout sommet est de degré pair, la suppression d'une arête ne détruit pas la connexité du graphe, mais il n'y a plus de cycle élémentaire.

☐ / ☐ S'il n'y a aucun cycle élémentaire de longueur supérieure ou égale à 3, alors le graphe contient forcément des arêtes multiples.

Fin.

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	
`ouvrirImage(nom)`	Ouvre le fichier *nom* et retourne l'image contenue dedans (par exemple `ouvrirImage("teapot.png")`).
`nouvelleImage(largeur,hauteur)`	Retourne une image de taille *largeur* × *hauteur*, initialement noire.
`ecrireImage(img, nom)`	Sauvegarde l'image *img* dans le fichier *nom*.
`largeurImage(img)` `hauteurImage(img)`	Récupère la largeur de *img*. Récupère la hauteur de *img*.
`colorierPixel(img, x, y, (r,g,b))`	Peint le pixel (x, y) dans l'image *img* de la couleur (r, g, b)
`(r,g,b) = couleurPixel(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`

L'argument `u` 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(G))` où `G` contient un graphe