Numéro d'anonymat :

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

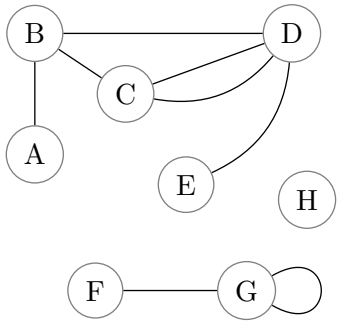
Examen du 21 janvier 2021
11h30-13h00

Le sujet comporte 6 exercices et 7 pages, dont une page d'annexe

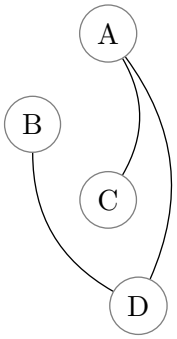
Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images et de graphes disponibles sont rappelées en annexe page 7. Vous pouvez détacher cette annexe pour plus de facilité.

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

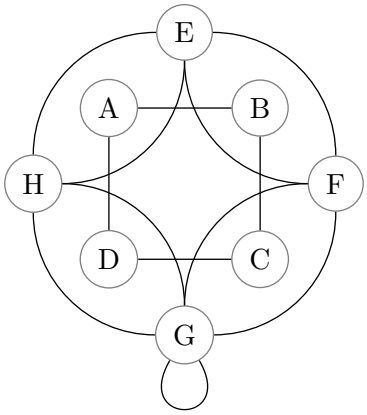
Graphe G_0



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 Lors d'une course à la voile, on établit journalièrement un relevé des skippers selon la distance qui reste à parcourir jusqu'au point d'arrivée. On suppose que ces distances sont sauvegardées dans une liste, mais qui n'est pas triée par ordre du classement. Ainsi, si l'on a la liste :
`relevesDesDistances=[131.70, 126.53, 130.56, 138.01, 126.34, 127.40, 131.23, 133.80]`
le premier skipper de la liste doit parcourir encore `131.70 mn` (milles nautiques), le deuxième `126.53 mn`, le troisième `130.56 mn`, et ainsi de suite. Le premier du classement doit parcourir encore `126.34 mn`, le deuxième du classement `126.53 mn`, etc.

1. Écrire une fonction `distanceMaximaleAparcourir(L)` qui renvoie la distance à parcourir par le dernier du classement.
   Sur l'exemple, `distanceMaximaleAparcourir(relevesDesDistances)` renvoie `138.01`.

2. Écrire une fonction `nbConcurrents(L, dist)` qui renvoie le nombre de skippers qui se trouvent à strictement moins de `dist` milles nautiques de l'arrivée.
   Sur l'exemple, l'appel `nbConcurrent(relevesDesDistances,126))` renvoie `0`,
   et `nbConcurrent(relevesDesDistances,127))` renvoie `2`.

Exercice 3 On considère la fonction suivante prenant comme paramètre un entier naturel n :

```
def mystere(n):
    s = 0
    while n > 0 :
        s = s + n % 10
        n = n // 10
    return s
```

1. Simuler l'exécution de `mystere(59564)` en complétant le tableau suivant :

n		
s		

2. Quelle est la valeur de `mystere(59564)` ?

3. De façon générale, que calcule la fonction `mystere(n)` ?

4. Le résidu d'un entier naturel est le chiffre obtenu en additionnant tous les chiffres du nombre initial, puis en additionnant les chiffres du nombre obtenu, et ainsi de suite jusqu'à l'obtention d'un nombre à un seul chiffre.

   Par exemple, le résidu du nombre 59564 est 2 car $5+9+5+6+4=29$, puis $2+9=11$, puis $1+1=2$.

   En faisant appel à la fonction `mystere` ci-dessus, écrire une fonction `residu(n)` qui retourne le résidu du nombre entier `n`.

Exercice 4 On souhaite tracer un "ruban" vertical au milieu d'une image. Pour cela on écrira une fonction `dessinerRuban(img, epaisseur, c)` prenant en paramètres :

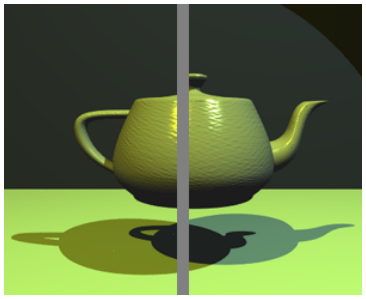
— une image `img` ;

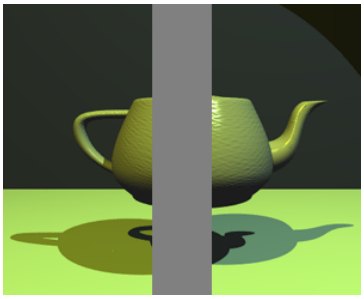
— un entier supposé positif `epaisseur` ;

— c, une couleur dans le modèle `RGB`.

Cette fonction doit dessiner un "ruban" vertical de couleur `c` dont les deux bords se trouveront à égale distance (à un pixel près) de la ligne qui sépare verticalement en deux l'image. La largeur du "ruban" est définie par la valeur de `epaisseur` que l'on suppose inférieure ou égale à la largeur de l'image.

Par exemple, l'image ci-dessous à gauche montre le résultat de l'exécution de l'appel à
`theiere = ouvrirImage("teapot.png")`
`dessinerRuban(theiere, 10, (128, 128, 128))`
et l'image à droite montre le résultat de l'exécution de l'appel à
`theiere = ouvrirImage("teapot.png")`
`dessinerRuban(theiere, 50, (128, 128, 128))`





Écrire en Python la fonction `dessinerRuban(img, epaisseur, c)`.

Exercice 5 Marquage

1. Écrire une fonction `auMoinsUnVoisinMarque(s)` qui prend en paramètre un sommet `s` d'un graphe et qui renvoie `True` si le sommet `s` a au moins un voisin marqué, et qui renvoie `False` sinon.

2. En vous aidant de `auMoinsUnVoisinMarque(s)`, écrire une fonction `sommetsNonMarquesVoisinsDAuMoinsUnSommetMarque(G)` qui prend en paramètre un graphe `G` et qui renvoie `True` si chaque sommet non marqué a au moins un voisin marqué, et qui renvoie `False` sinon.

Exercice 6 Un graphe G est dit *complet* si chaque sommet de G est voisin avec tous les autres sommets de G.

1. Soit G un graphe à n sommets. Montrer que, si G est complet, le degré de chaque sommet de G est supérieur ou égal à $n-1$.

2. Dessiner un exemple de graphe G à 3 sommets, dont tous les sommets sont de degré ≥ 2, mais qui n'est pas complet.
   Aide : ce graphe peut avoir des boucles.

3. Soit G un graphe à n sommets qui est *simple* c'est à dire qu'il n'a pas d'arête qui boucle sur un sommet, ni d'arête multiple c'est-à-dire pas d'arêtes $e \neq f$ qui ont les mêmes extrémités. Montrer que G est complet si et seulement si le degré de chaque sommet de G est égal à $n-1$.

4. En utilisant ce résultat, écrire une fonction `completS(G)` qui prend en argument un graphe `G` supposé simple, et retourne `True` si `G` est complet, `False` si `G` est incomplet.

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`
`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.