Numéro d'anonymat :

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

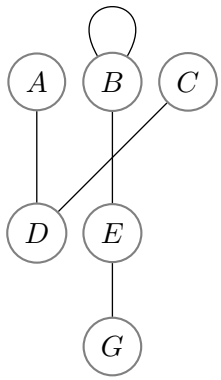
Examen du 9 janvier 2017
9h-10h30

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

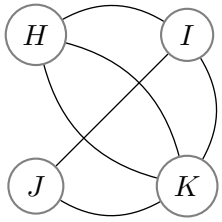
Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images et de graphes disponibles sont rappelées en annexe page 9. Vous pouvez détacher cette annexe pour plus de facilité.

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

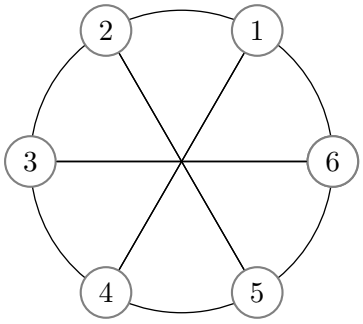
Graphe G_0



Graphe G_1



Graphe G_2



1. Remplir le tableau ci-dessous en n'oubliant pas de justifier votre réponse lorsque c'est nécessaire.

	G_0	G_1	G_2
Est-il connexe ? (si non, justifier)			
Donner un sommet de degré maximal et préciser son degré.			

Exercice 2 On considère la fonction suivante :

```
def mystere(n):
    if n == 0:
        return 0
    k = -1
    while n > 0 :
        ch = n%10
        if  ch%2 == 0:
            k = ch
        n = n // 10
    return k
```

1. Simuler l'exécution de `mystere(3276)` en complétant le tableau suivant :

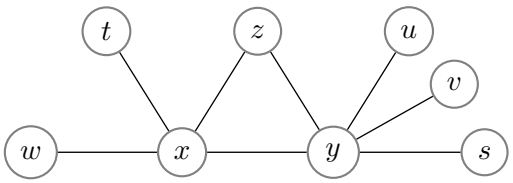
ch		
k		
n		

2. Que vaut `mystere(3276)` ?
3. Que vaut `mystere(511)` ?
4. D'une manière générale, indiquer ce que retourne `mystere(n)` en fonction de l'entier naturel n.

Exercice 3 Ecrire une fonction `auMoinsUnElementCommun(L1, L2)` qui retourne `True` si les listes d'entiers `L1` et `L2` ont au moins un élement en commun, et qui retourne `False` sinon.

Par exemple, les listes `[2, 15, 17]` et `[5, 32, 12]` n'ont aucun élément en commun ; les listes `[2, 15, 17]` et `[5, 32, 2]` ont un élément en commun.

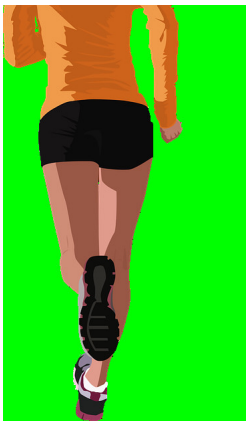
Exercice 4 Dans un graphe, une *feuille* est un sommet de degré 1. Par exemple, le graphe dessous possède cinq feuilles, dont deux adjacentes au sommet x (feuilles voisines de x) et trois adjacentes au sommet y.



1. Écrire une fonction `nbFeuillesAdjacentes(s)`, qui retourne le nombre de feuilles adjacentes à un sommet s.

2. Écrire une fonction `tousOntUneFeuille(G)`, qui teste si tous les sommets du graphe G de degré strictement supérieur à 1 ont au moins une feuille adjacente, en faisant appel à la fonction `nbFeuillesAdjacentes`.

Exercice 5 On dispose d'une photographie d'un personnage prise sur un fond vert (sur la photocopie le vert apparait en gris clair !). On souhaite incruster ce personnage sur un autre fond comme dans l'exemple suivant.



1. Ecrire une fonction `incruster (imagePersonnage, imageFond)` qui incruste l'image `imagePersonnage` dans l'image `imageFond`. Cette dernière image sera modifiée par la fonction. On place l'image du personnage en haut à gauche de l'image de fond (on fait correspondre leurs coins supérieurs gauches). On suppose que le fond vert qui ne doit pas apparaître dans l'image résultat est composé uniquement de pixels de couleur (0, 255, 0). On suppose également que l'image de fond est assez grande pour contenir toute l'image à incruster.

Exercice 6 1. Ecrivez une fonction `diagonaleCarre(img, c)` qui prend en paramètre une image carrée (c'est à dire telle que la hauteur de l'image est égale à sa largeur) et qui colorie avec la couleur `c` les pixels de la diagonale qui part du coin supérieur gauche.

2. Ecrivez une fonction `triangleInferieurGauche(img,c)` qui prend en paramètre une image carrée et qui colorie les pixels du triangle inférieur gauche (diagonale incluse) avec la couleur `c`.

3. On suppose disposer d'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)`. Ecrivez une fonction `drapeau(img)` qui dessine le drapeau ci-dessous (les bandes grises et noires sont de largeur `taille`/4, plus ou moins l'erreur d'arrondi). N.B. : pour écrire très simplement cette fonction `drapeau`, vous pouvez utiliser, en plus de la fonction `rectanglePlein`, la fonction de la question précédente même si vous n'avez pas réussi à l'écrire.

Exercice 7

1. Donner une formule reliant les degrés des sommets et le nombre d'arêtes d'un graphe.

2. Un graphe peut-il avoir un seul sommet de degré impair ? Justifier.

3. Quelle propriété mathématique (simple) possède le nombre de sommets de degré impair d'un graphe ? Justifier.

4. On suppose qu'un graphe G comporte exactement deux sommets de degré impair. Par raisonnement par l'absurde, démontrer que ces deux sommets sont dans la même composante connexe.

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 *large* × *haut*, initialement noire.
`(largeur,hauteur) = img.size`	Récupère la largeur et 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)
`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`