Numéro d'anonymat :

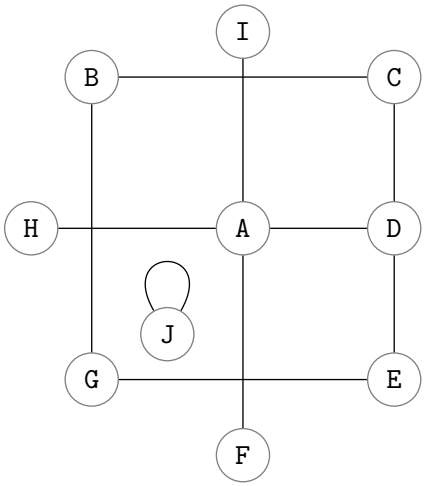
Université de Bordeaux — Examen du 19 janvier 2024
Département Licence, Informatique 1, **code UE 4TPU146U/4TPU147U** — 9h00-10h30

Le sujet comporte 7 exercices et 13 pages, dont une page d'annexe

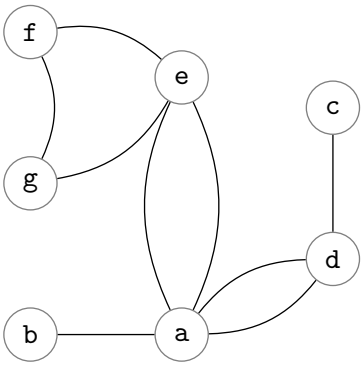
Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images et de graphes disponibles sont rappelées en annexe page 13. Vous pouvez détacher cette annexe pour plus de facilité.

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

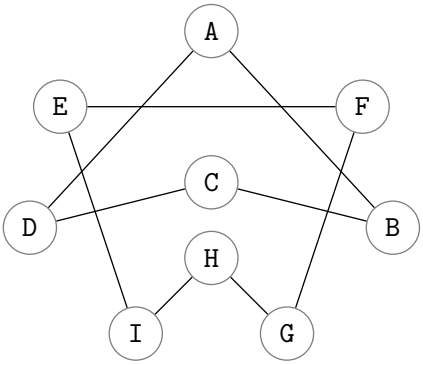
Graphe G_0



Graphe G_1



Graphe G_2



Remplir le tableau ci-dessous.

	G_0	G_1	G_2
Est-il connexe ? Justifier			
Donner un sommet de degré maximal et préciser son degré.			
Donner un cycle élémentaire de longueur maximale. Il est possible de donner une simple liste de sommets.			

Exercice 2 – Logique de base

1. Soit l'énoncé (P2) suivant : $\exists x \in \mathbb{R}, \forall y \in \mathbb{R}, y^2 > x$.
   Dire s'il est vrai (dans ce cas le démontrer par preuve directe) ou faux (dans ce cas donner un contre-exemple).

   Ecrire sa négation :

2. Soit l'énoncé (P3) suivant : "**si** n^2 **est impair, alors** n **est impair**" où n est un entier naturel.
   Donner la contraposée de l'énoncé (P3) :

   Démontrer cette contraposée. On pourra utiliser le fait qu'un nombre pair peut se traduire par $2p$ où p est un entier naturel.

   Donner la négation de l'énoncé (P3) et sa valeur de vérité (vraie ou fausse).

   Donner la réciproque de l'énoncé (P3).

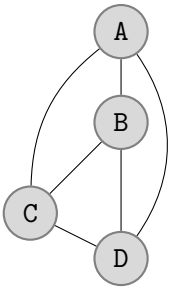
Exercice 3 - Fonctions python sur les graphes

Pour rappel, un graphe est dit **simple** s'il n'a ni boucle, ni arête multiple.

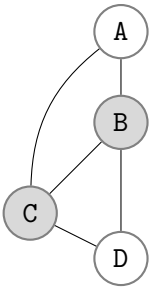
Un sommet d'un graphe simple est dit **universel** s'il est relié à **tous** les autres sommets du graphe.

Un graphe simple est dit **complet** si **tous** ses sommets sont universels.

Un exemple de graphe complet est donné **Figure 1a** et un exemple de graphe non complet est donné **Figure 1b**.



(a) Le graphe complet à 4 sommets



(b) Un graphe non complet à 4 sommets

FIGURE 1 – *Les sommets universels sont dessinés légèrement en gris, pour mieux les distinguer des autres.*

Dans la suite de cet exercice, les graphes passés en paramètre des fonctions sont supposés simples.

Le but de l'exercice est de décider si un graphe simple donné, ayant au moins un sommet, est complet.

1. Pour un graphe simple à n sommets, quel est le degré d'un sommet universel ?

2. Écrire une fonction `sommetEstUniversel(G:graphe, s:sommet) -> bool` qui prend en paramètre un graphe simple G et un sommet s de G et qui retourne `True` si s est un sommet universel et `False` sinon.

3. Écrire une fonction `nbSommetsUniversels(G:graphe) -> int` qui prend en paramètre un graphe simple G et qui retourne le nombre de sommets universels de G (utilisez `sommetEstUniversel` même si vous n'avez pas écrite)

4. Écrire une fonction `grapheEstComplet(G:graphe) -> bool` qui prend en paramètre un graphe simple G et qui retourne `True` si G est un graphe complet et `False` sinon.

Exercice 4 Fonctions mystere Soient les fonctions `mystere1` et `mystere2` suivantes :

```
def mystere1(img1, img2, y1, y2, dx):
    larg=largeurImage(img1)
    for x in range (larg):
        for y in range(y1, y2):
            (r,g,b)=couleurPixel(img1, x, y)
            colorierPixel(img2, (x+dx)%larg, y, (r,g,b))

def mystere2(img1, img2, y1, y2, dx):
    larg=largeurImage(img1)
    for x in range (larg):
        for y in range(y1, y2):
            (r,g,b)=couleurPixel(img1, x, y)
            colorierPixel(img2, (x-dx)%larg, y, (r,g,b))
```

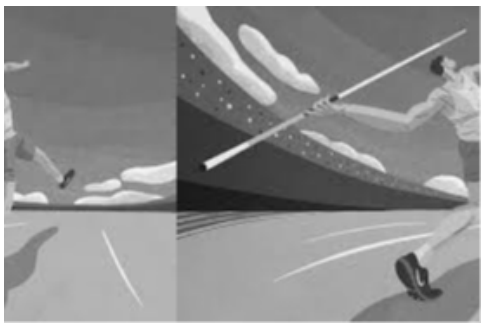
Le fichier `personnage.jpeg` contient l'image suivante :

La largeur de l'image est de 275 pixels et la hauteur de 183 pixels.
Soient les lignes de code suivantes :

```
img=ouvrirImage("personnage.jpeg")
larg=largeurImage(img)
haut=hauteurImage(img)
nouvImg1=nouvelleImage(larg,haut)
nouvImg2=nouvelleImage(larg,haut)
mystere1(img,nouvImg1,0,haut,100)
mystere2(img,nouvImg2,0,haut,100)
afficherImage(nouvImg1)
afficherImage(nouvImg2)
```

1. Parmi les 5 images ci-dessous, quelle image s'affiche pour nouvImg1 ?(Indiquez la lettre)

2. Parmi les 5 images ci-dessous, quelle image s'affiche pour nouvImg2 ?(Indiquez la lettre)



(a)

(b)

(c)

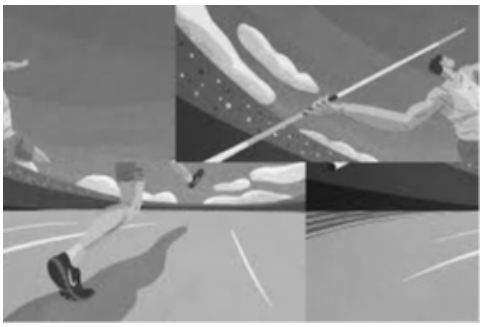
(d)

(e)

3. Dire en une phrase synthétique et en justifiant la réponse ce que fait la fonction `mystere1` suite à l'appel des lignes de code précédentes (on ne veut pas de paraphrase du code !) :

4. Dire en une phrase synthétique et en justifiant la réponse ce que fait la fonction `mystere2` suite à l'appel des lignes de code précédentes (on ne veut pas de paraphrase du code !) :

Soit l'image suivante :



5. Ecrire les lignes de code permettant d'afficher cette image, en appelant les fonctions `mystere1` et `mystere2` (en commençant par ouvrir l'image du fichier `personnage.jpeg` comme précédemment) :

Exercice 5 - Images

Écrire une fonction `repliquerCouleursBande (img:image, d:int)` qui modifie la couleur des pixels de l'image `img` sur une bande de hauteur `d` pixels en haut de l'image et sur une bande de hauteur `d` pixels en bas de l'image de façon à obtenir une sorte de cadre :

— dans la colonne d'abscisse `x` de l'image, les `d` pixels de la bande en haut de l'image seront coloriés avec la couleur du pixel de plus grande ordonnée dans cette bande.

— dans la colonne d'abscisse `x` de l'image, les `d` pixels de la bande en bas de l'image seront coloriés avec la couleur du pixel de plus petite ordonnée dans cette bande.

Dans la figure ci-dessous, l'image de droite (b) est obtenue en appelant la fonction `repliquerCouleursBande (img:image, d:int)` sur l'image de gauche (a) (de dimensions 512×320) et en donnant la valeur 40 au paramètre `d`.

(a) (b)

FIGURE 3 – Manipulation d'images

Exercice 6 – Marche aléatoire dans un graphe

Une marche aléatoire est une chaîne où les sommets sont choisis aléatoirement. Pour cela il suffit de partir d'un sommet donné, puis choisir aléatoirement un voisin de ce sommet. Si ce sommet n'a pas déjà été visité, on le rajoute à notre chaîne et on recommence tant que l'on rencontre un sommet non visité.

1. Choisissez les bons mots dans cette phrase :

   Lors d'une marche aléatoire, le nombre de sommets parcourus est *connu/inconnu*, je vais préférer utiliser une boucle *for/while*.

2. Quelle fonction du module `bibgraphes` pourrait-on utiliser pour savoir si un sommet a déjà été visité ?

3. On souhaite réaliser une fonction `longueurMarcheAlea(s: sommet) -> int` qui, prenant un sommet `s`, parcourt le graphe jusqu'à retomber sur un sommet déjà visité et renvoie la longueur de la chaîne parcourue. (*Exemple : Le parcours* `[A,E,F,E]` *renvoie l'entier* `3`) . (On suppose que tous les sommets sont de degré au moins 1, et tous blancs et démarqués).
   Écrivez la fonction `longueurMarcheAlea(s: sommet) -> int`. On pourra utiliser la fonction `randrange` ou `elementAleatoireListe`

Exercice 7 - Démonstration sur les graphes

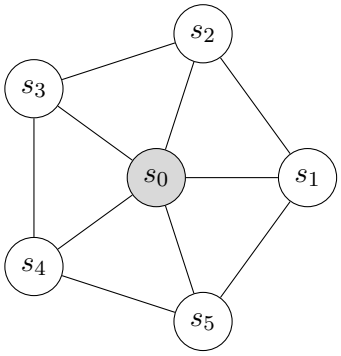
Important : Dans cet exercice nous ne considérerons que des **graphes simples** (c'est-à-dire sans boucle, ni arêtes multiples).

On définit une roue au rang n comme un graphe à $n+1$ sommets (avec $n \geq 4$) obtenu à partir :

— d'un cycle élémentaire de longueur n,

— et d'un sommet central voisin de tous les sommets du cycle.

On notera un tel graphe, une roue R_n. Vous trouverez l'exemple d'une roue R_5 ci-dessous. Le sommet central s_0 a été dessiné légèrement en gris, pour mieux le distinguer des autres sommets.



Dans toute la suite on considérera que $n \geq 4$.

1. Quel est le nombre d'arêtes d'une roue R_n ?

2. Quels sont les degrés possibles des sommets d'une roue R_n ?

3. Calculer la somme des degrés d'une roue R_n

4. En déduire à l'aide de la formule de la « *poignée de mains* » que vous réécrirez, le nombre d'arêtes d'une roue R_n

5. Dessiner un graphe à 7 sommets tel que chaque sommets soit de degré 2 et que le graphe soit connexe.

6. Dessiner un graphe à 7 sommets tel que chaque sommet soit de degré 2 et que le graphe soit non connexe.

Soit le théorème :

Théorème 1 : Si G est un graphe simple connexe et que chaque sommet est de degré 2, alors G contient un et un seul cycle élémentaire.

On se propose de démontrer ce théorème. On sépare la démonstration en deux parties :

a) Montrer que le graphe G contient au moins un cycle élémentaire.

a) Montrer que le graphe G contient au plus un cycle élémentaire.

Montrer pour chacune des propositions suivantes si elle est VRAIE ou FAUSSE. Si elle est vraie vous la démontrerez, si elle est fausse vous exhiberez un contre-exemple. Pour ce faire, vous pourrez utiliser le théorème précédent (même si vous n'avez pas su le démontrer).

Énoncé 1 – Soit G un graphe simple avec n sommets de degré 3 et 1 sommet de degré n (avec $n \geq 4$), alors G est connexe

Énoncé 2 – Soit G un graphe simple avec n sommets de degré 3 et 1 sommet de degré n (avec $n \geq 4$), alors G est une roue R_n

Énoncé 3 – Soit G un graphe simple avec n sommets de degré 3 et 1 sommet de degré n (avec $n \geq 4$). Si en supprimant le sommet de degré n (avec ses arêtes incidentes) et que le graphe résultant reste connexe, alors G est une roue R_n

Fin.

Annexe détachable

Annexe - Vous pouvez détacher cette annexe pour plus de facilité

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).

`randrange (debut:int, fin:int) -> int`	retourne un nombre entier aléatoire compris entre `debut` (inclus) et `fin` (exclus).
`elementAleatoireListe (L:list)`	retourne un élément pioché aléatoirement dans la liste `L`.

Ci-dessous, `img` est une image	
`ouvrirImage(nom:str) -> image`	Ouvre le fichier *nom* et retourne l'image contenue dedans (par exemple `img = ouvrirImage("teapot.png")`).
`nouvelleImage(large:int, haut:int) -> image`	Retourne une image de taille $large \times haut$, initialement noire.
`afficherImage(img:image)`	Affiche l'image *img*.
`L = largeurImage(img)` `H = hauteurImage(img)`	Récupère la largeur de *img*. Récupère la hauteur de *img*.
`colorierPixel(img:image, x:int, y:int, (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:graphe) -> list`	retourne la *liste* des *sommets* de `G`
`nbSommets(G:graphe) -> int`	retourne le *nombre* de *sommets* de `G`
`sommetNom(G:graphe, etiquette:str) -> sommet`	retourne le *sommet* de `G` désigné par son *nom* (`etiquette`). Exemple : `s = sommetNom(Europe, 'Italie')`

L'argument `s` est un sommet	
`listeVoisins(s:sommet) -> list`	retourne la *liste* des *voisins* de `s`
`degre(s:sommet) -> int`	retourne le *degré* de `s`
`nomSommet(s:sommet) -> str`	retourne le *nom* (étiquette) de `s`
`colorierSommet(s:sommet, c:str)`	colorie `s` avec la couleur `c`. Exemples de couleurs : `'red'`, `'green'`, `'blue'`, `'white'`, `'cyan'`, `'yellow'`
`couleurSommet(s:sommet) -> str`	retourne la *couleur* de `s`
`marquerSommet(s:sommet)` `demarquerSommet(s:sommet)`	marque le sommet `s` démarque le sommet `s`
`estMarqueSommet(s:sommet) -> bool`	retourne `True` si `s` est marqué, `False` sinon

Remarque : les couleurs possibles des sommets font partie d'une liste prédéfinie ; par exemple, "white", "black", "red", "green", "blue", "yellow", ...