Numéro d'anonymat :

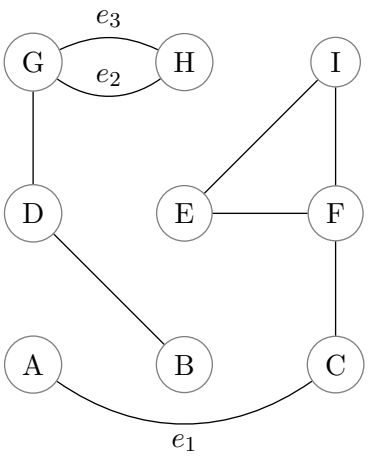
Université de Bordeaux — Examen du 25 janvier 2023
Département Licence, Informatique 1, **code UE 4TPU146U/4TPU147U** — 9h00-10h30

Le sujet comporte 6 exercices et 9 pages, dont une page d'annexe

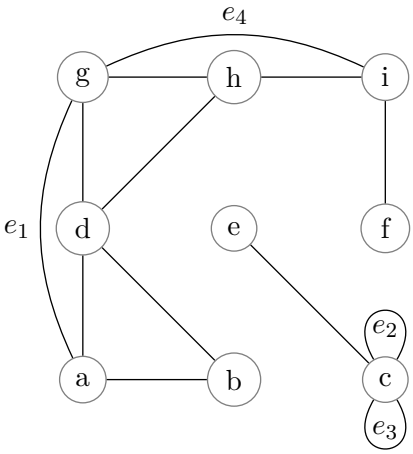
Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images et de graphes disponibles sont rappelées en annexe page 9. Vous pouvez détacher cette annexe pour plus de facilité.

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

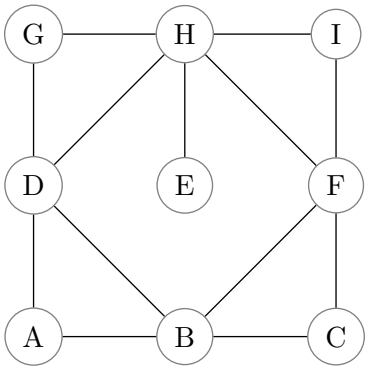
Graphe G_0



Graphe G_1



Graphe G_2



Remplir le tableau ci-dessous.

	G_0	G_1	G_2
Est-il connexe ? (si non, 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, sauf s'il y a ambiguïté.			

Exercice 2 – Logique de base

Partie 1

Soit l'énoncé `P1` : « ***Les multiples de deux sont des nombres pairs*** ». Cet énoncé est VRAI, il n'y a pas besoin de le démontrer.

1. Donner sa réciproque.

La réciproque de `P1` est-elle VRAIE ou FAUSSE ? (Cocher la bonne réponse)
☐ VRAIE
☐ FAUSSE (dans ce cas donner ci-dessous un contre-exemple)

Écrire la contraposée de `P1`.

Soit l'énoncé `P2` : « ***Les multiples de quatre sont des nombres pairs*** ». Cet énoncé est VRAI, il n'y a pas besoin de le démontrer.

2. Donner sa réciproque.

La réciproque de `P2` est-elle VRAIE ou FAUSSE ? (Cocher la bonne réponse)
☐ VRAIE
☐ FAUSSE (dans ce cas donner ci-dessous un contre-exemple)

Écrire la contraposée de `P2`.

Partie 2

Soient I un intervalle de $\mathbb{R}$ non vide et $f : I \to \mathbb{R}$ une fonction à valeurs réelles définie sur I.

3. Soit l'énoncé $\exists M \in \mathbb{R}, \forall x \in I, |f(x)| \leq M$. On suppose que $I = [4, 8]$ et $f(x) = 2x$ pour tout $x \in I$.
   Dire si l'énoncé est VRAI (dans ce cas le démontrer) ou FAUX (dans ce cas donner un contre-exemple) :

   Exprimer à l'aide des quantificateurs, la négation de l'énoncé : $\exists M \in \mathbb{R}, \forall x \in I, |f(x)| \leq M$

4. Soit l'énoncé $\forall y \in \mathbb{R}, \exists x \in I, f(x) = y$. On suppose que $I = [3, 6]$ et $f(x) = x^2$ pour tout $x \in I$.
   Dire si l'énoncé est VRAI (dans ce cas le démontrer) ou FAUX (dans ce cas donner un contre-exemple) :

   Exprimer à l'aide des quantificateurs, la négation de l'énoncé : $\forall y \in \mathbb{R}, \exists x \in I, f(x) = y$

Exercice 3 Soit le code ci-dessous :

```
def mystere (valeurs:list) -> list:
    histo = [0] * 6
    for valeur in valeurs:
        histo[valeur] = histo[valeur] + 1
    return histo

valeursA = [3, 4, 5, 2, 3, 4, 5, 0, 3]
valeursB = [1, 2, 1, 2, 1, 5, 0, 0]
```

1. Que retournent les appels suivants :

(a) `mystere(valeursA)`

(b) `mystere(valeursB)`

2. Expliquer en quelques mots ce que retourne l'appel `mystere(valeurs)`.

Exercice 4 – Lancers de dés

On rappelle que l'appel de fonction `randrange(debut, fin)` retourne un entier aléatoire compris entre l'entier `debut` et l'entier `fin-1` inclus.

1. Écrire une fonction `lancer() -> int` qui ne prend pas d'argument, et renvoie la valeur aléatoire d'un lancer de dé à 6 faces. Il vous est demandé de faire appel à la fonction `randrange`.

2. On lance deux dés à 6 faces simultanément, de manière répétée jusqu'à ce que les deux valeurs obtenues, x, y, soient égales à deux prédéfinies, a, b, peu importe l'ordre (x peut être égal à a et y à b ou x égal à b et y à a). On cherche à compter le nombre de tentatives avant que cela ne se réalise.
   Écrire une fonction `nbLancers(a:int, b:int) -> int`, qui, étant données deux valeurs `a` et `b` entre 1 et 6, renvoie le nombre de tentatives (double-lancers) effectuées avant d'obtenir un `a` et un `b`.

3. On réalise n fois l'expérience précédente, et l'on souhaite obtenir la moyenne des nombres de lancers nécessaires.

   Écrire une fonction `moyenneNbLancers(a:int, b:int, n:int) -> float`, qui prend en paramètres deux valeurs `a`, `b` définies comme précédemment, une valeur `n` correspondant au nombre de répétitions que l'on souhaite, et qui renvoie la moyenne du nombre de lancers nécessaires pour obtenir un `a` et un `b` avec deux lancers simultanés.

   Il vous est **demandé** de faire un appel à la fonction précédente, et de ne pas en recopier le code.
   Exemple : si l'on prend `n = 4` et que les résultats de l'expérience sont respectivement 10, 14, 20 et 16 lancers, le résultat renvoyé par la fonction sera le flottant `15.0`.

Exercice 5 - Images

Dans cet exercice, on suppose que `img1` et `img2` sont deux images de même taille.

1. Écrire une fonction

   `copierCollerQuadrant(img1:image, img2:image, x:int, y:int, dstx:int, dsty:int)`

   qui copie un seul quadrant de `img1` dont les coordonnées du premier pixel sont `x` et `y` et le colle dans un autre quadrant de `img2` dont les coordonnées du premier pixel sont `dstx`, `dsty`. On précise que le premier pixel d'un quadrant représente le pixel situé au coin haut-gauche de ce quadrant. Ainsi, le résultat de l'appel suivant `copierCollerQuadrant(img1, img2, 0, 0, l//2, 0)` avec comme premier paramètre l'image de la théière et comme deuxième paramètre une image noire de même taille produira le résultat illustré dans la Figure 1b (page suivante).

2. Écrire une fonction `mosaique(img1:image, img2:image)` qui copie tous les quadrants de `img1` et les colle dans d'autres quadrants de `img2` pour ainsi créer une mosaïque. Chaque quadrant sera déplacé vers le quadrant voisin en suivant une rotation dans le sens des aiguilles d'une montre. Ceci est représenté dans la Figure 1c.

   NB : Il vous est demandé pour l'écriture de la fonction `mosaique` de faire appel à la fonction `copierCollerQuadrant` précédemment créée autant de fois que nécessaire et de veiller à bien paramétrer chaque appel à la fonction.

3. On veut à présent tester et visualiser l'image produite par la fonction `mosaique`. Écrire la séquence d'instructions correspondante, de façon à ce que le premier paramètre de cette fonction est l'image de la théière récupérée à partir du fichier nommé "teapot.png" et le deuxième paramètre est une nouvelle image de même taille.

(a) (b) (c)

FIGURE 1 – Manipulation d'images
(a) L'image de la théière `img1` avec les numéros des quadrants.
(b) L'image générée après l'exécution de l'appel `copierCollerQuadrant(img1,img2,0,0,l//2,0)`.
(c) L'image générée après l'exécution de l'appel `mosaique(img1, img2)` avec les numéros des quadrants après leurs déplacements.

Exercice 6 - Graphes

On définit le *poids* d'une arête d'un graphe comme la somme des degrés des deux sommets que l'arête relie. Par exemple, le poids de l'arête A−B du graphe G_2 de l'exercice 1 est de 6, puisque le sommet A est de degré 2 et le sommet B est de degré 4.

1. Identifier l'arête (les arêtes s'il y en a plusieurs) de poids maximum dans le graphe G_2 de l'exercice 1. Préciser son (leur) poids. Vous identifierez le nom d'une arête par le nom des sommets à chacune de ses extrémités comme par exemple l'arête A−B.

2. Dessiner un exemple de graphe ayant 5 sommets au total, dont quelques-uns de degré 2 et d'autres de degré 3, dans lequel toutes les arêtes sont de poids 5.

3. Écrire une fonction `poidsMaximum(G)` qui calcule et retourne le poids le plus grand d'une arête dans un graphe donné `G`.

4. Une arête de poids maximum est-elle toujours incidente à au moins un sommet de degré maximum ? Si oui, justifier, si non, donner un contre-exemple.

5. Prouver que dans un graphe quelconque, la somme totale des poids de toutes les arêtes du graphe est égale à la somme des carrés des degrés de tous les sommets du graphe.
Indice : Étudier précisément la contribution d'un sommet fixe à la somme des poids des arêtes.

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, fin)`	retourne un nombre entier aléatoire compris entre `debut` (inclus) et `fin` (exclus).

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", ...