Nom: Prénom: Groupe:

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

Devoir surveillé du 24 novembre 2020
9h00 – 10h00

Le sujet comporte 5 exercices et 7 pages, dont une page d'annexe

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

Écrivez votre nom et prénom sur chaque feuille pour éviter qu'elles se perdent.

Exercice 1 Considérons la fonction `mystere` suivante prenant en paramètre un entier `n`.

```
def mystere(n):
    u = 2
    v = 1
    for i in range(n):
        w = v
        v = u
        u = v - w + 1
    return u
```

1. Simulez l'exécution de l'appel `mystere(3)` en complétant le tableau suivant qui montre l'évolution des variables :

`u`		
`v`		
`w`		
`i`		

2. Que retourne l'appel `mystere(3)` ?

Exercice 2 On souhaite écrire une fonction `multiple3positifs` prenant en unique paramètre d'entrée une liste d'entiers relatifs `L` supposée non vide et qui renvoie `True` si tous les éléments de `L` sont des multiples de `3` positifs, `False` sinon.

1. Soit la liste `Ltest = [6, 5, 9, -2, -1]`. Que renvoie `multiple3positifs(Ltest)` ?

2. Écrire la portion de code définissant la fonction `multiple3positifs`.

Exercice 3 Écrire la fonction `nbElementsDansIntervalle(L, debut, fin)` qui prend en paramètre une liste `L` d'entiers, 2 entiers `debut` et `fin`, avec `debut` $\leq$ `fin`, et qui compte et retourne le nombre d'éléments `i` de `L` tel que `i`$\in$ [`debut`,`fin`].

Par exemple, le résultat de l'appel `nbElementsDansIntervalle([12,7,3,1,8,6],4,7)` serait 2 car $7 \in [4, 7]$ et $6 \in [4, 7]$.

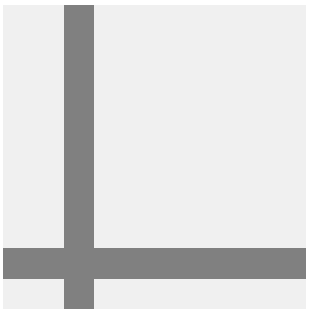
Nom: Prénom: Groupe:

Exercice 4 Croisillon

1. Nous voulons écrire une fonction `tracerCroisillon(img, x1, y1, c)` qui prend en paramètres :
   - une image `img`
   - x1, une abscisse valide dans l'image `img`
   - y1, une ordonnée valide dans l'image `img`
   - c, une couleur dans le modèle `RGB`

   Cette fonction doit dessiner dans l'image `img` une ligne et une colonne, de couleur `c`. La ligne verticale est tracée à l'abscisse `x1`, la ligne horizontale est tracée à l'ordonnée `y1`.

   Par exemple, l'image ci-dessous montre le résultat de l'exécution de l'appel à
   `tracerCroisillon(carreGris, 2, 8, (128, 128, 128))`
   pour une image `carreGris` qui était initialement uniformément gris clair de largeur 10 et de hauteur 10.



   Écrire en Python la fonction `tracerCroisillon(img, x1, y1, c)`.

2. On souhaite maintenant écrire une fonction
`tracerCroisillonPixelsAlternes(img, x1, y1, c1, c2)`
qui tracera un croisillon en alternant deux couleurs.

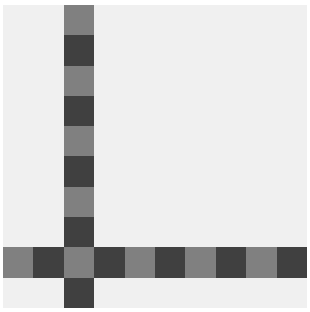
Elle prend en paramètre une image `img`, une valeur d'abscisse valide `x1` et une valeur d'ordonnée valide `y1`. Les paramètres `c1` et `c2` sont deux couleurs dans le modèle `RGB`. Comme précédement cette fonction dessine dans l'image `img` une colonne à l'abscisse `x1` et une ligne à l'ordonnée `y1`, mais à la différence de la fonction précédente, les pixels ne sont pas coloriés avec une même couleur.

La règle pour colorier chaque pixel de ces deux lignes est la suivante :

— il est colorié avec la couleur `c1` si la somme de ses deux coordonnées est un nombre pair

— il est colorié avec la couleur `c2` sinon

Par exemple, l'image ci-dessous montre le résultat de l'exécution de l'appel à
`tracerCroisillonPixelsAlternes(carreGris, 2, 8, (128, 128, 128), (64,64,64))`
pour une image `carreGris` qui était initialement uniformément gris clair de largeur 10 et de hauteur 10.



Écrire en Python la fonction `tracerCroisillonPixelsAlternes(img, x1, y1, c1, c2)`.

Nom: Prénom: Groupe:

Exercice 5 Triangle

Nous voulons écrire une fonction `triangleVersLaDroite(img, x1, y1, y2, c)` qui prend en paramètres :

— une image `img`,

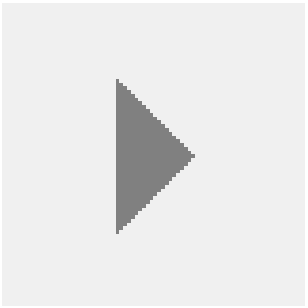
— `x1`, une abscisse valide dans l'image `img`

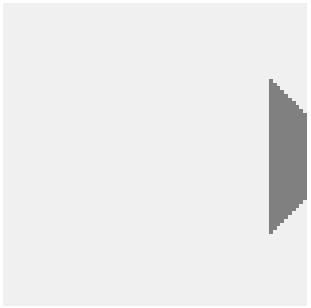
— `y1` et `y2`, deux ordonnées valides dans l'image `img`, avec `y1` $\leq$ `y2`

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

Cette fonction doit dessiner dans l'image `img` une succession de segments de droite verticaux de la façon suivante : le premier segment va du pixel de coordonnées `(x1,y1)` au pixel de cordoonnées `(x1,y2)` (compris), le segment suivant va du pixel de coordonnées `(x1+1,y1+1)` au pixel de cordoonnées `(x1+1, y2-1)`, le suivant va du pixel de coordonnées `(x1+2,y1+2)` au pixel de cordoonnées `(x1+2,y2-2)`, et ainsi de suite, autant que possible. Cela forme ainsi un triangle dirigé vers la droite.

Les images ci-dessous montrent le résultat de l'exécution de l'appel à cette fonction sur une image `carreGris` qui était initialement uniformément gris clair de largeur 80 et de hauteur 80. L'image de gauche correspond à l'appel :
`triangleVersLaDroite(carreGris, 30, 20, 60, (128, 128, 128))`
et celle de droite à l'appel :
`triangleVersLaDroite(carreGris, 70, 20, 60, (128, 128, 128))`





Écrire en Python la fonction `triangleVersLaDroite(img, x1, y1, y2, c)`.

Nom: Prénom: Groupe:

Annexe : voici un rappel des principales fonctions disponibles pour manipuler les images (à vous de voir celles qui sont utiles pour ce devoir).

Ci-dessous, `img` est une image	
`img = ouvrirImage(nom)`	Ouvre le fichier *nom* et retourne l'image contenue dedans (par exemple `ouvrirImage("teapot.png")`).
`img = nouvelleImage(largeur,hauteur)`	Retourne une image de taille $largeur \times hauteur$, initialement noire.
`L = largeurImage(img)` `H = 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)