Nom: Prénom: Groupe:

Université de Bordeaux — Devoir surveillé du 07 novembre 2018
Département Licence, **4TPM101U** — 11h30-13h

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 disponibles sont rappelées en annexe page 7. Vous pouvez détacher cette annexe pour plus de facilité.

Exercice 1 Considérons la fonction `mystere` suivante prenant en paramètre une liste `L` composée de 0 ou de 1 :

```
def mystere(L):
    c=0
    j=1
    for i in L :
        c=c+i*j
        j=j*2
    return c
```

1. Simulez l'exécution de l'appel `mystere([1,1,0,1])` en complétant le tableau suivant qui montre l'évolution des variables `c`, `j` et `i` :

`c`		
`j`		
`i`		

2. Que retourne l'appel `mystere([1,1,0,1])` ?

3. Quelle liste L permet d'obtenir la valeur 14 lorsque l'on appelle `mystere(L)` ?

4. Écrire une fonction `laListeEstBonne` qui prend en paramètre une liste L composée de chiffres quelconques et retourne True si la liste L ne comporte que des 0 ou des 1, False dans le cas contraire.

Exercice 2 Pixels négatifs

L'objectif de cet exercice est d'appliquer un filtre sur la moitié droite d'une image `img`. Plus précisément, nous souhaitons inverser en négatif les couleurs de chaque pixel de l'image `img` : si un pixel a la couleur `(r,g,b)`, alors sa nouvelle couleur est modifiée en `(255-r,255-g,255-b)`. Voici un exemple :

image initiale

moitié droite inversée

1. Écrire une fonction `inverser(img)` qui inverse la couleur des pixels de la moitié droite de l'image `img`.

Nom: Prénom: Groupe:

Exercice 3 *Amplitude* d'une liste de notes.

Soit `L` une liste de nombres réels représentant des notes entre 0 et 20.

1. Écrire une fonction `minimumListe(L)` qui calcule et retourne le minimum de la liste (on attire l'attention sur le fait que les notes dans `L` sont toutes inférieures ou égales à 20). Par exemple, `minimumListe([9,10.5,17,15.5,11,5,11])` retournera `5`.

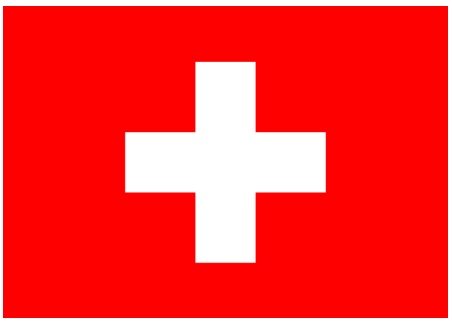
2. Écrire une fonction `amplitude(L)` qui, en parcourant une seule fois la liste, calcule et retourne la différence entre le maximum et le minimum des notes dans `L`. Par exemple, `amplitude([9,10.5,17,15.5,11,5,11])` retournera `12` (c'est à dire `17-5`).

Exercice 4 Écrire les fonctions suivantes prenant en paramètre une liste de nombres `L`.

1. Écrire une fonction `positionPremierNegatif(L)` qui retourne la position du premier nombre négatif dans `L`, numérotée à partir de 0, ou `-1` si `L` n'en contient aucun.
   Par exemple, `positionPremierNegatif([-3,4,-2,1])` retournera `0`.

2. Écrire une fonction `positionDernierNegatif(L)` qui retourne la position du dernier nombre négatif dans `L`, numérotée à partir de 0, ou `-1` si `L` n'en contient aucun.
   Par exemple, `positionDernierNegatif([-3,4,-2,1])` retournera `2`.

Exercice 5 Calcul largeur d'une croix



On dispose d'une image du drapeau suisse, où l'on a une croix blanche au sein d'un rectangle rouge. On ne connait pas la taille de la croix, l'image ci-dessus n'est qu'un exemple. On souhaite calculer la largeur de la croix, en détectant le bord extrême gauche et le bord extrême droit de la croix.

Nom: Prénom: Groupe:

1. Écrire une fonction `trouverMinX(image)` qui retourne le minimum des coordonnées X des pixels blancs de l'image `image` (donc le bord extrême gauche)

2. De même, écrire une fonction `trouverMaxX(image)` qui retourne le maximum des coordonnées X des pixels blancs de l'image `image` (donc le bord extrême droit)

3. En utilisant les fonctions `trouverMinX(image)` et `trouverMaxX(image)`, écrire une fonction `calculerLargeur(image)` qui retourne la largeur de la croix, i.e. l'écartement entre le bord extrême gauche et le bord extrême droit. Il n'est pas nécessaire d'avoir traité les deux questions précédentes pour traiter celle-ci.

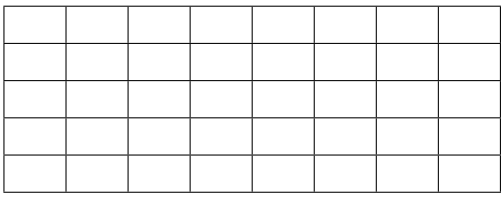
Exercice 6 On considère le code de la fonction `triangleSuperieur(img, c)` qui colorie avec la couleur `c` tous les pixels de l'image `img` au dessus de la diagonale, diagonale comprise.

```
def triangleSuperieur(img, c):
    largeur = largeurImage(img)
    hauteur = hauteurImage(img)
    if largeur > hauteur:
        for x in range(largeur):
            for y in range(0, ((x * (hauteur - 1)) // (largeur - 1)) + 1):
                colorierPixel(img, x, y, c)
    else:
        for y in range(hauteur):
            for x in range((y * (largeur - 1)) // (hauteur - 1), largeur):
                colorierPixel(img, x, y, c)
```

1. On crée une image `monImage` et l'on appelle la fonction `triangleSuperieur` sur `monImage` ainsi :

   ```
   monImage = nouvelleImage(8, 5)
   triangleSuperieur(monImage, (0,0,255))
   ```

   Voici une représentation de l'image `monImage`. Cochez les pixels se retrouvant colorés par l'appel à la fonction `triangleSuperieur`.



2. En vous inspirant de la fonction `triangleSuperieur`, écrire la fonction `triangleInferieur(img, c)` qui colorie avec la couleur `c` tous les pixels de l'image `img` au dessous de la diagonale, diagonale comprise.

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).

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 $largeur \times hauteur$, 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*