Nom: Prénom: Groupe:

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

Devoir surveillé du 26 novembre 2019
11h30 – 12h30

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

Aucun document n'est autorisé – Toutes les fonctions de manipulation d'images disponibles sont rappelées en annexe page 5. Vous pouvez détacher cette annexe pour plus de facilité.

Exercice 1 Considérons la fonction `mystere` suivante prenant en paramètre une liste `L` de nombres réels positifs :

```
def mystere(L):
    cpt = 0
    cptMax = 0
    for i in L :
        if i > 0:
            cpt = cpt + 1
        else:
            if cpt > cptMax :
                cptMax = cpt
            cpt = 0
    if cpt > cptMax :
        cptMax = cpt
    return cptMax

precipitations = [9.9, 0, 5.5, 7.8, 0, 10.5, 0]
```

1. Simulez l'exécution de l'appel `mystere(precipitations)` en complétant le tableau suivant montrant l'évolution des variables `i`, `cpt` et `cptMax` à la fin de chaque tour de boucle `for` :

`i`		
`cpt`		
`cptMax`		

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

3. Que retourne l'appel `mystere([0, 10.3, 0, 0, 7.9, 8.9, 6.3])` ?

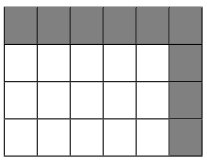
4. De façon générale, comment interpréter la valeur retournée par la fonction `mystere(L)`, pour une liste `L` de nombres réels positifs ?

5. Écrire la fonction `sansPluie(L)`, qui prend pour paramètre une liste de nombres réels positifs `L`, et qui retourne `True` si cette liste ne contient aucune valeur strictement positive, `False` sinon. Par exemple `sansPluie(precipitations)` retourne `False` alors que `sansPluie([0, 0, 0])` retourne `True`.

Exercice 2 Écrire la fonction `sommeMultiplesDe3(L)`, qui prend en parametre une liste d'entiers `L`, et qui retourne la somme de tous les éléments multiples de 3 se trouvant avant un éventuel `-1`. Par exemple, le résultat de l'appel `sommeMultiplesDe3([1, 3, 4, 3, 3, 1, 2, 2, 6])` serait `15` car la liste ne contient pas de -1, tandis que le résultat de l'appel `sommeMultiplesDe3([1, 3, 4, 27, 18, -1, 2, 2, 6])` serait `48`.

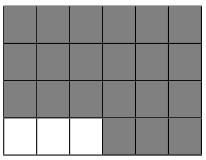
Exercice 3

1. Écrire la fonction `coinUnPixel(img)` qui trace en rouge une bordure d'un pixel en haut et à droite de l'image `img`. Par exemple, avec une image de 6×4 pixels, les pixels à colorier sont ceux marqués en gris ci-dessous :



Nom: Prénom: Groupe:

2. Écrire la fonction `coin(img,p)` qui trace une bordure de `p` pixels en rouge en haut et à droite de l'image `img`. Par exemple, pour une image de 6×4 pixels, l'appel à `bordure(img,3)` doit colorier les pixels marqués en gris ci-dessous :



Exercice 4 <u>Lecture de code</u> – **Dans cet exercice on supposera que toutes les images traitées par les fonctions sont de dimensions (largeur et hauteur) paires.**

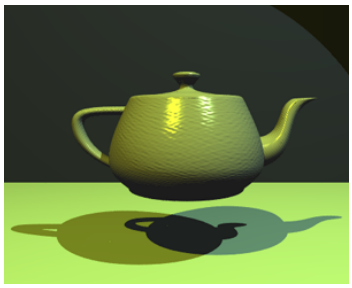
On considère les deux fonctions suivantes :

```
def traitement1(img):
    H = hauteurImage(img)
    L = largeurImage(img)
    for x in range(L//2):
        for y in range(H):
            c = couleurPixel(img, x, y)
            colorierPixel(img, L//2 + x, y, c)
```

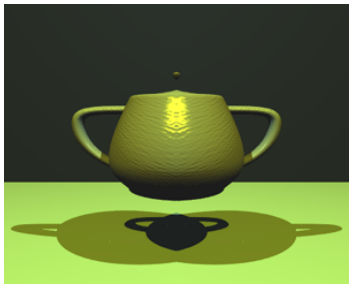
et

```
def traitement2(img):
    H = hauteurImage(img)
    L = largeurImage(img)
    for y in range(H):
        for x in range(L//2):
            c1 = couleurPixel(img, x, y)
            c2 = couleurPixel(img, L//2 + x, y)
            colorierPixel(img, x, y, c2)
            colorierPixel(img, L//2 + x, y, c1)
```

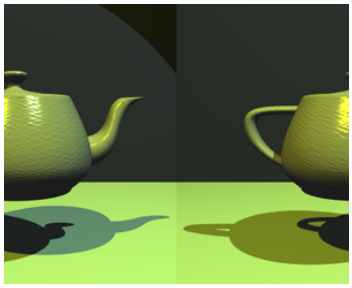
On suppose que la variable `theiere` contient l'image (de dimensions paires) suivante :



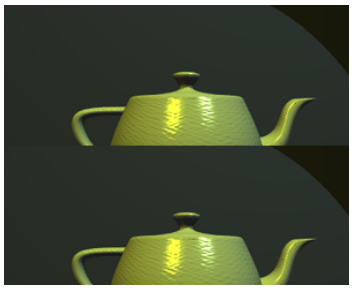
Indiquer quelle image, parmi les six suivantes, est produite par les fonctions `traitement1` et `traitement2` si celles-ci sont appelées sur l'image de la variable `theiere` d'origine.



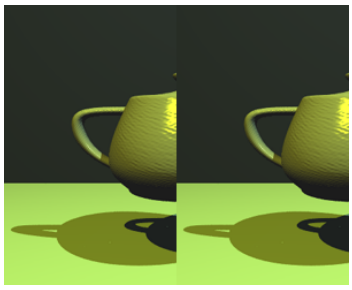
A



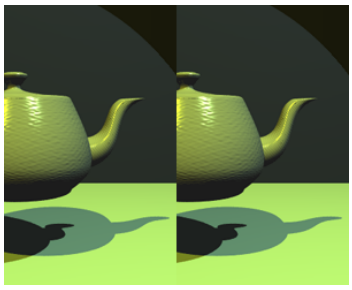
B



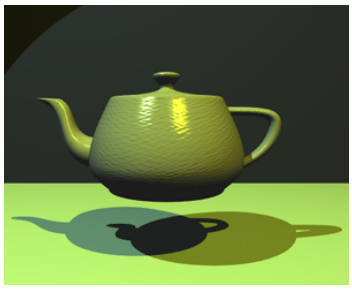
C



D



E



F

Nom: Prénom: Groupe:

`traitement1(theiere)` produit l'image

`traitement2(theiere)` produit l'image

Exercice 5 Image et son négatif

Avec le modèle `RGB`, le négatif de la couleur (r, g, b) est $(255 - r, 255 - g, 255 - b)$.

Écrivez une fonction `estLeNegatif(img1,img2)` qui compare deux images de même taille `img1` et `img2` et retourne `True` si `img2` est le négatif de `img1` et `False` sinon.

Par exemple si `img1` correspond à l'image de gauche ci-dessous et `img2` à l'image de droite la fonction `estLeNegatif(img1,img2)` retournera `True`

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.
`ecrireImage(img, nom)`	Sauvegarde l'image *img* dans le fichier *nom*.
`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)
`(r,g,b) = couleurPixel(img, x, y)`	Retourne la couleur du pixel (x, y) dans l'image *img*