Détection et extraction de panneaux

Nom	
Prénom	
Classe	

Question : 1	**Note**
Ce sont des array à 3 dimensions : Lignes, Colonnes, Triplets	

Question : 2	**Note**
Entiers codés sur 8 bits	

Question : 3	**Note**
3 entiers par pixel, soit 3 fois 1 octet (8 bits) Soit 3 octets	

Question : 4	**Note**
Lignes x Colonnes x 3 octets 600 x 800 x 3 1 440 000 octets 1.4 Mo	

Question : 5	**Note**
t = 0.000001 T = 600*800*3*t Temps de traitement estimé: 1.44 s	

Question : 6	Note

```
import matplotlib.pyplot as plt
def Affiche(fig,im):
    plt.figure(fig)
    plt.imshow(im)
    plt.axis('off')
    plt.show()
```

Question : 7	Note

```
Nom_Image = "Image_1.bmp"
Image = plt.imread(Nom_Image)
Image = Image[:,:,:3]
Affiche(1,Image)
```

Question : 8	Note

```
import numpy as np
```

Question : 9	Note

```
def NB(im,alpha):
    Nl,Nc = im.shape[0:2]
    imnb = np.copy(im)
    for c in range(Nc):
        for l in range(Nl):
            R,G,B = im[l,c]
            R,G,B = float(R),float(G,),float(B)
            Cond_1 = R > alpha*(G+B)
            Cond_2 = R - max(G,B) > 2*alpha*(max(G,B)-min(G,B))
            if Cond_1 and Cond_2:
                Pix = [0,0,0]
            else:
                Pix = [255,255,255]
            imnb[l,c] = Pix
    return imnb
```

Question : 10	Note

```
alpha = 0.75
Image_NB = NB(Image,alpha)
Affiche(2,Image_NB)
```

Question : 11	Note

l et c les nombres de lignes et colonnes
O(l*c)

Question : 12	Note

```
def Seuil(imng,k):
    Nl,Nc = imng.shape[0:2]
    imnb = np.copy(imng)
    for c in range(Nc):
        for l in range(Nl):
            N,_,_ = imng[l,c]
            if N <= k:
                Pix = [0,0,0]
            else:
                Pix = [255,255,255]
            imnb[l,c] = Pix
    return imnb
```

Question : 13	Note

```
K = (1/9)*np.array([[1,1,1],[1,1,1],[1,1,1]])
Image_NG = Convolution(Image_NB,K)
Image_NB = Seuil(Image_NG,127)
Affiche(3,Image_NB)
```

Question : 14		Note

```
def Liste_voisins(l,c,Nl,Nc):
    vg = l,max(0,c-1)
    vd = l,min(Nc-1,c+1)
    vb = min(Nl-1,l+1),c
    vh = max(0,l-1),c
    return [vg,vd,vb,vh]
```

```
def Liste_voisins(l,c,Nl,Nc):
    GDBH = [[l,c-
1],[l,c+1],[l+1,c],[l-1,c]]
    for i in range(len(GDBH)):
        L,C = GDBH[i]
        if not (0<=L<=Nl-1 and
0<=C<=Nc-1):
            GDBH[i] = [l,c]
    return GDBH
```

Question : 15	Note

```
def Explorer(imnb,l,c,T,k):
    Nl,Nc,_ = imnb.shape
    Pile = [[l,c]]
    Val = imnb[l,c,0] # Inutile de mettre dans while
    while len(Pile) > 0:
        l,c = Pile.pop()
        T[l,c] = k
        Lv = Liste_voisins(l,c,Nl,Nc)
        for v in Lv:
            lv,cv = v
            T_v = T[lv,cv]
            Val_v = imnb[lv,cv,0]
            if T_v==-1 and Val_v == Val:
                Pile.append([lv,cv])
```

Question : 16	Note

```
def Zones(imnb):
    Nl,Nc = imnb.shape[0:2]
    T = -np.ones([Nl,Nc],dtype='int64')
    k = 0
    for l in range(Nl):
        for c in range(Nc):
            if T[l,c] == -1:
                Explorer(imnb,l,c,T,k)
                k += 1
    return T
```

Question : 17	Note

```
Table = Zones(Image_NB)
Affiche(4,Table)
Nb_Zones = np.amax(Table) + 1
print("Nombre de zones: ",Nb_Zones)
```

Question : 18		Note
Zone 1	`Nl-1,Nc-1,0,0`	
Zone 2	`T[l,c]`	
Zone 3	`(Ml*Taille+l)/(Taille+1)`	
Zone 4	`(Mc*Taille+c)/(Taille+1)`	
Zone 5	`imnb[l,c,0]`	
Zone 6	`l_min = l`	
Zone 7	`l_max = l`	
Zone 8	`c_min = c`	
Zone 9	`c_max = c`	
Code	`Donnees_Zones = Donnees(Table,Nb_Zones,Image_NB)`	

Question : 19	Note

```
def LC_Bilin(l,c,LO,LC):
    O1,_,_,O4 = LO
    C1,C2,C3,C4 = LC
    l12,c13 = O1
    l34,c24 = O4
    Vl = [l34-l,l-l12]
    Mat = [[C1,C2],[C3,C4]]
    Vc = [c24-c,c-c13]
    Res = Prod_MV(Mat,Vc)
    Res = Prod_VV(Res,Vl)
    Cst = 1/((l34-l12)*(c24-c13))
    Res = [int(round(t*Cst,0)) for t in Res]
    return Res
```

Question : 20	Note

```
def Coins(im):
    Nl,Nc = im.shape[0:2]
    hg = [0,0]
    hd = [0,Nc-1]
    bg = [Nl-1,0]
    bd = [Nl-1,Nc-1]
    return [hg,hd,bg,bd]
```

Question : 21	Note

```
def Resize(im,dim):
    Nl,Nc = dim
    im_rec = 255*np.ones([Nl,Nc,3],dtype='uint8')
    LO = Coins(im)
    LC = Coins(im_rec)
    for l in range(Nl):
        for c in range(Nc):
            L,C = LC_Bilin(l,c,LC,LO)
            im_rec[l,c] = im[L,C]
    return im_rec
```

Question : 22	Note

```
def Distance_uv(u,v):
    n = len(u)
    Dst = 0
    for i in range(n):
        di = u[i]-v[i]
        Dst += di**2
    Dst = Dst**(1/2)
    return Dst
```

Question : 23	Note

```
def Analyse(im):
    Nl,Nc = im.shape[0:2]
    L_RGB = []
    for l in range(Nl):
        for c in range(Nc):
            R,G,B = im[l,c]
            R = float(R)
            G = float(G)
            B = float(B)
            L_RGB += [R,G,B]
    return L_RGB
```

Question : 24	Note

```
def Recadrage_k(im,LD,T,k):
    _,_,_,_,l_min,c_min,l_max,c_max = LD[k]
    dl,dc = l_max - l_min + 1,c_max - c_min + 1
    im_rec = 255*np.ones((dl,dc,3),dtype='uint8')
    for l in range(dl):
        for c in range(dc):
            ll,cc = l+l_min,c+c_min
            if T[ll,cc]==k:
                im_rec[l,c] = im[ll,cc]
    return im_rec
```

Question : 25	Note

```
from math import sqrt
def Etude_Motif(imnb,imc,T,LD,a):
    imc = Resize(imc,(a,a))
    imc_nb = NB(imc,alpha)
    L_RGB_c_nb = Analyse(imc_nb)
    Res = []
    for k in range(len(LD)):
        im_loc_nb = Recadrage_k(imnb,LD,T,k)
        im_loc_nb = Resize(im_loc_nb,(a,a))
        L_RGB_im_loc_nb = Analyse(im_loc_nb)
        d = Distance_uv(L_RGB_c_nb,L_RGB_im_loc_nb)
        dn = d/(255*a*sqrt(3))
        Res.append([dn,k])
    Res.sort()
    return Res
```

Question : 26	Note

```
Im_Cercle = plt.imread("Cercle.bmp")
a = 100
Distances = Etude_Motif(Image_NB,Im_Cercle,Table,Donnees_Zones,a)
```

Question : 27	Note

```
def Selection(Ldst,dmax,LD):
    Res = []
    for d,k in Ldst:
        if d <= dmax:
            Res.append(LD[k])
    return Res
```

Question : 28	Note

```
dmax = 0.4
```

Question : 29	Note

```
Zones_Sel = Selection(Distances,dmax,Donnees_Zones)
```

Question : 30	Note

```
def Recadrage(im,l_min,l_max,c_min,c_max):
    dl,dc = l_max - l_min + 1,c_max - c_min + 1
    im_rec = np.zeros((dl,dc,3),dtype='uint8')
    for l in range(dl):
        for c in range(dc):
            ll,cc = l+l_min,c+c_min
            im_rec[l,c] = im[ll,cc]
    return im_rec
```

Question : 31	Note

```
ind = 1
for zone in Zones_Sel:
    _,_,_,_,l_min,c_min,l_max,c_max = zone
    Panneau = Recadrage(Image,l_min,l_max,c_min,c_max)
    Panneau = Resize(Panneau,(100,100))
    Affiche(100+ind,Panneau)
    plt.imsave("Panneau "+str(ind)+".bmp",Panneau)
    ind += 1
```