Name: **Muhammad Aqeel Afzal**

Roll. No: **i190650**

Section: **8A**

Assignment: **Report 1st**

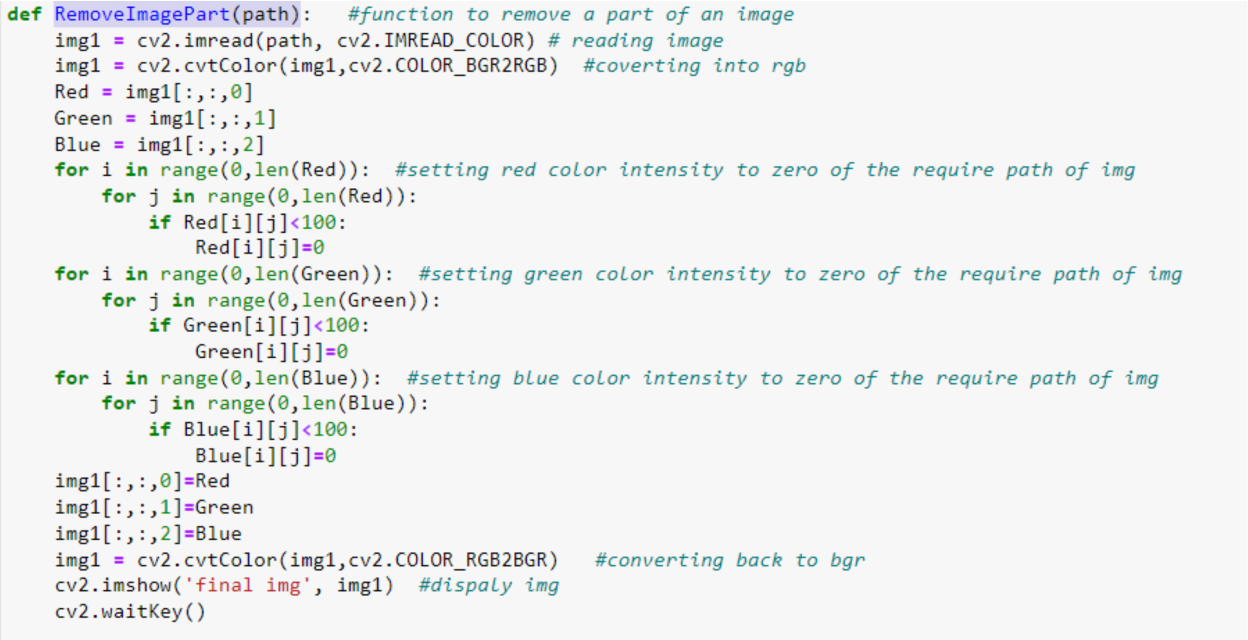
Submitted To: **Dr. Akhtar Jamil**

Report

Question 01:

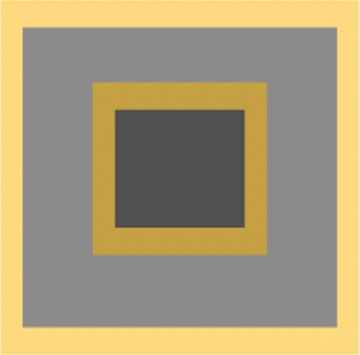
RemoveImagePart(path) function takes an image as input and removes the desired part(middle) of the image and then displays it.

Code:

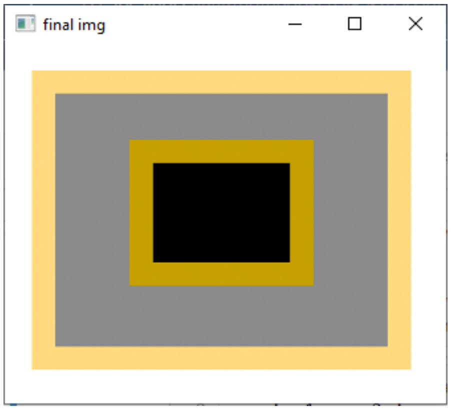


```
def RemoveImagePart(path):   #function to remove a part of an image
    img1 = cv2.imread(path, cv2.IMREAD_COLOR) # reading image
    img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)  #coverting into rgb
    Red = img1[:,:,0]
    Green = img1[:,:,1]
    Blue = img1[:,:,2]
    for i in range(0,len(Red)):  #setting red color intensity to zero of the require path of img
        for j in range(0,len(Red)):
            if Red[i][j]<100:
                Red[i][j]=0
    for i in range(0,len(Green)):  #setting green color intensity to zero of the require path of img
        for j in range(0,len(Green)):
            if Green[i][j]<100:
                Green[i][j]=0
    for i in range(0,len(Blue)):  #setting blue color intensity to zero of the require path of img
        for j in range(0,len(Blue)):
            if Blue[i][j]<100:
                Blue[i][j]=0
    img1[:,:,0]=Red
    img1[:,:,1]=Green
    img1[:,:,2]=Blue
    img1 = cv2.cvtColor(img1,cv2.COLOR_RGB2BGR)   #converting back to bgr
    cv2.imshow('final img', img1)  #dispaly img
    cv2.waitKey()
```

Input image:

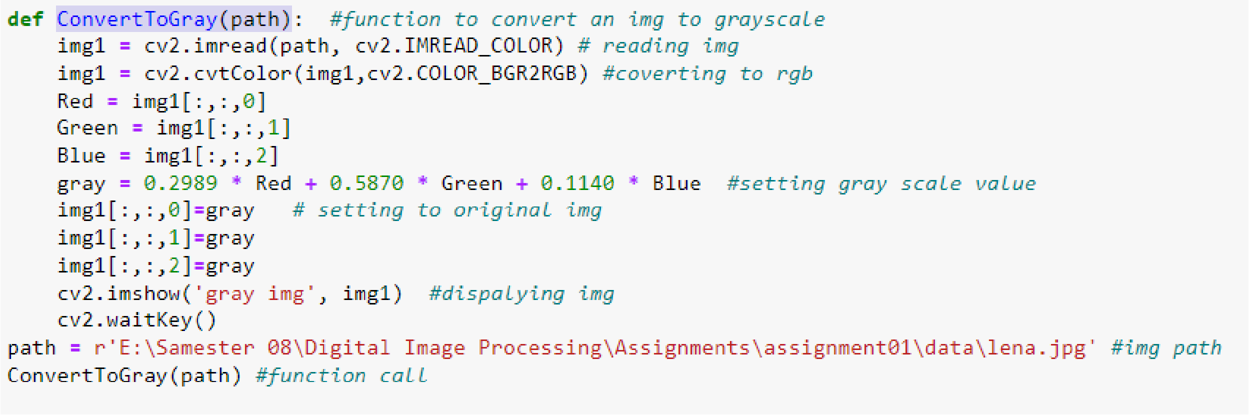


Output image:



Question 02:

ConvertToGray(path) function takes an image as input and converts the image to grayscale and then displays it.

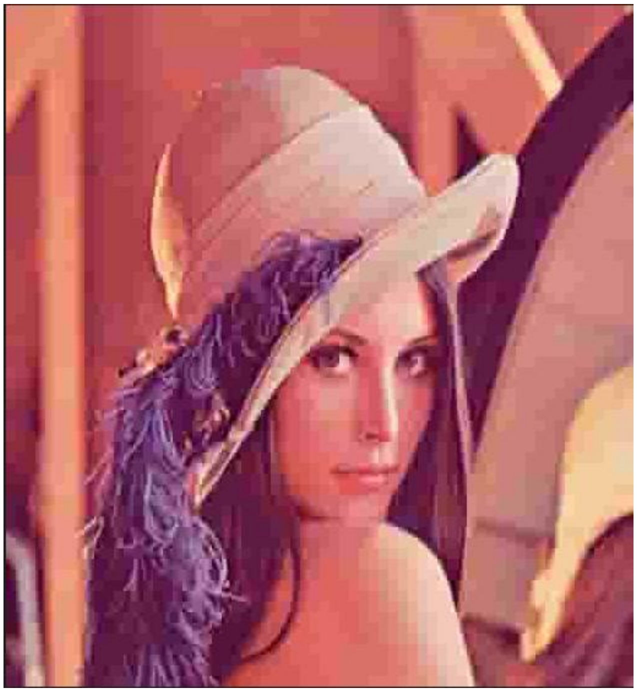


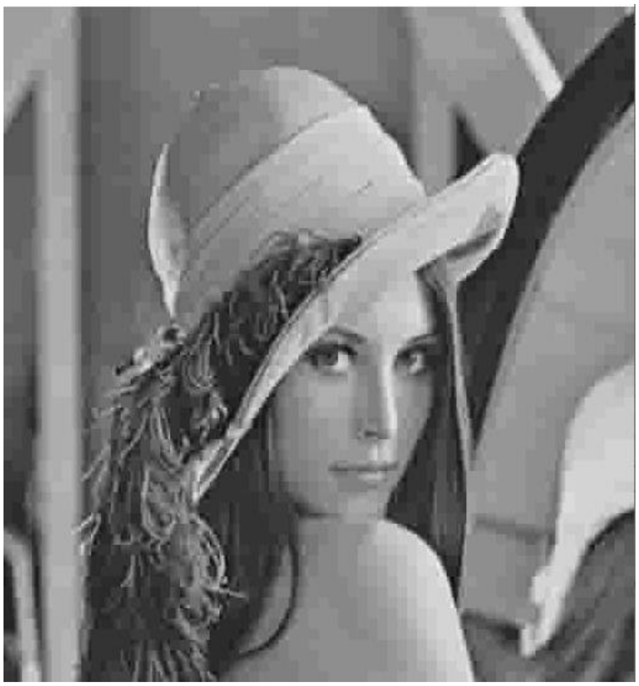
```python
def ConvertToGray(path):  #function to convert an img to grayscale
    img1 = cv2.imread(path, cv2.IMREAD_COLOR) # reading img
    img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB) #coverting to rgb
    Red = img1[:,:,0]
    Green = img1[:,:,1]
    Blue = img1[:,:,2]
    gray = 0.2989 * Red + 0.5870 * Green + 0.1140 * Blue  #setting gray scale value
    img1[:,:,0]=gray   # setting to original img
    img1[:,:,1]=gray
    img1[:,:,2]=gray
    cv2.imshow('gray img', img1)  #dispalying img
    cv2.waitKey()
path = r'E:\Samester 08\Digital Image Processing\Assignments\assignment01\data\lena.jpg' #img path
ConvertToGray(path) #function call
```

Code:

Input image:

Output image:

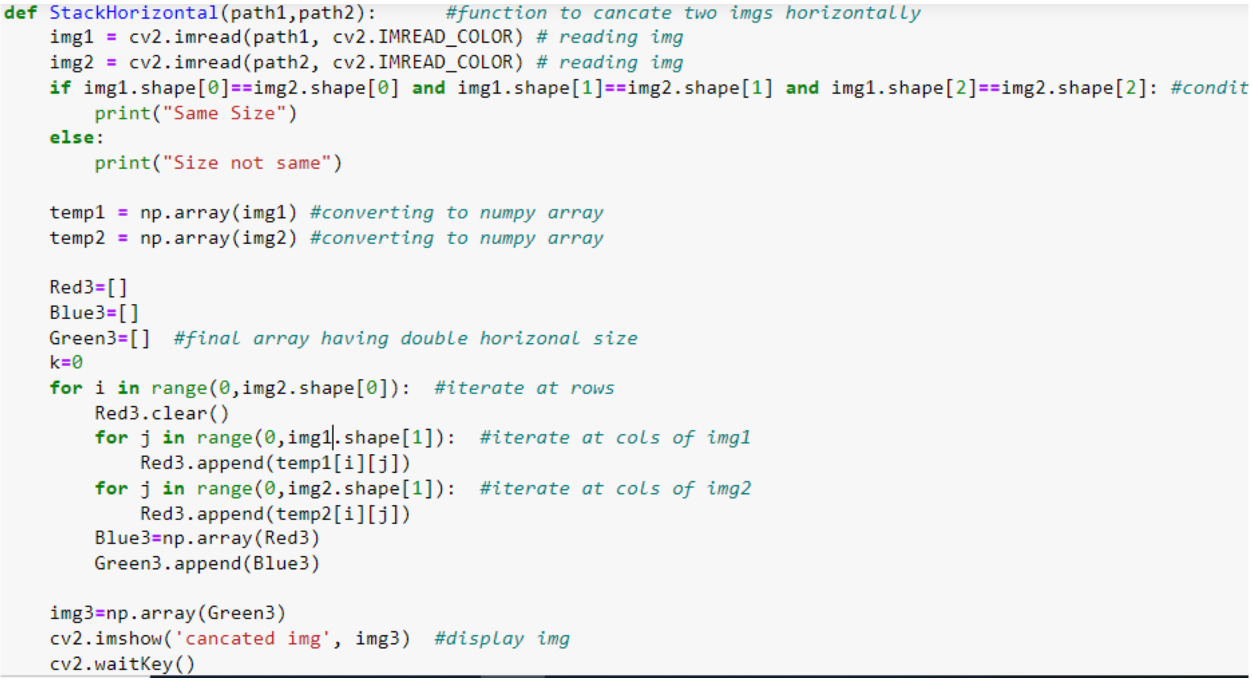




Question 03:

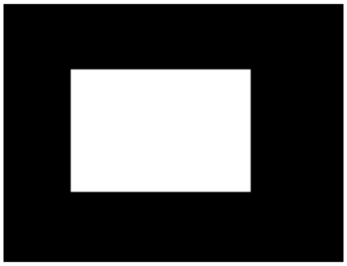
StackHorizontal(path1,path2) function takes two images as input and concatenates them, and then displays them.

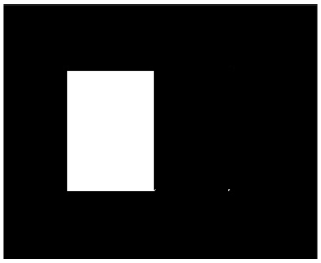
Code:



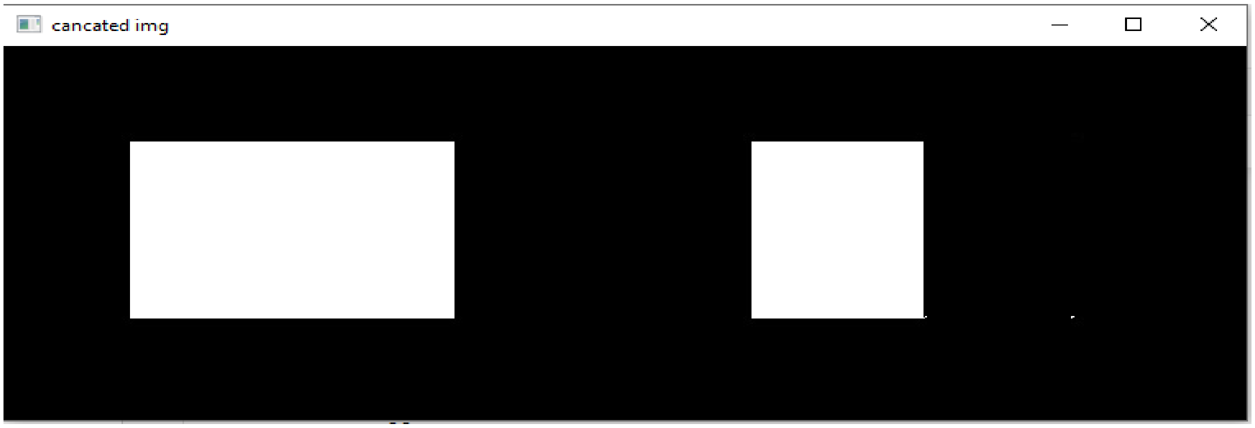
```
def StackHorizontal(path1,path2):      #function to cancate two imgs horizontally
    img1 = cv2.imread(path1, cv2.IMREAD_COLOR) # reading img
    img2 = cv2.imread(path2, cv2.IMREAD_COLOR) # reading img
    if img1.shape[0]==img2.shape[0] and img1.shape[1]==img2.shape[1] and img1.shape[2]==img2.shape[2]: #condit
        print("Same Size")
    else:
        print("Size not same")

    temp1 = np.array(img1) #converting to numpy array
    temp2 = np.array(img2) #converting to numpy array

    Red3=[]
    Blue3=[]
    Green3=[]  #final array having double horizonal size
    k=0
    for i in range(0,img2.shape[0]):  #iterate at rows
        Red3.clear()
        for j in range(0,img1.shape[1]):  #iterate at cols of img1
            Red3.append(temp1[i][j])
        for j in range(0,img2.shape[1]):  #iterate at cols of img2
            Red3.append(temp2[i][j])
        Blue3=np.array(Red3)
        Green3.append(Blue3)

    img3=np.array(Green3)
    cv2.imshow('cancated img', img3)  #display img
    cv2.waitKey()
```

Input images:





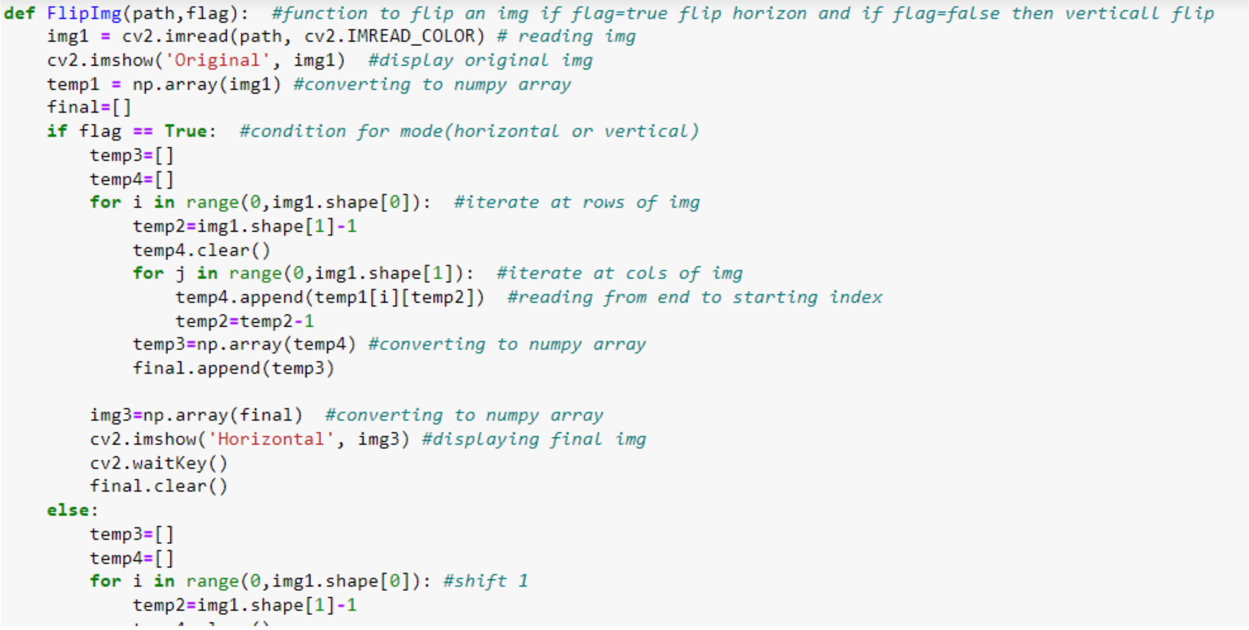
Output image:



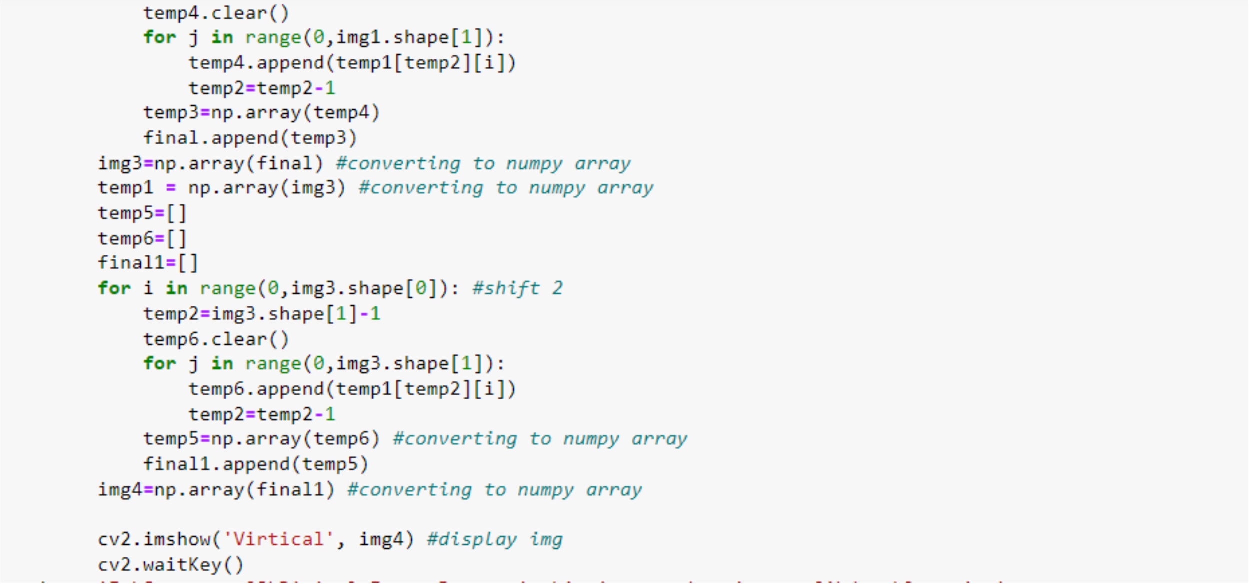
Question 04:

FlipImg(path, flag) function takes two arguments as input one as an image and the second as a flag if the flag is true then it will flip the image horizontally and if the flag is false then flip the image vertically and display the flipped image as well as the original image.

Code:

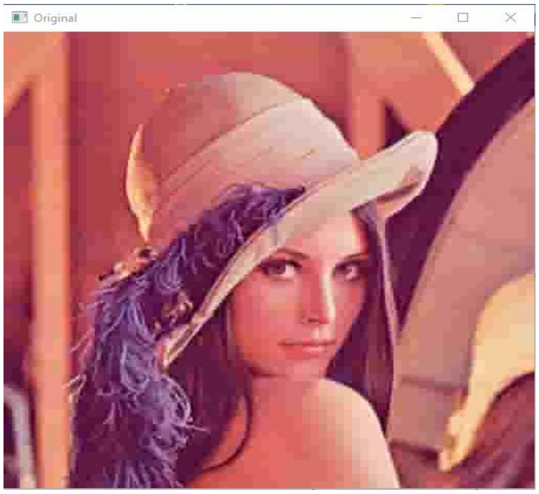


```
def FlipImg(path,flag):  #function to flip an img if flag=true flip horizon and if flag=false then verticall flip
    img1 = cv2.imread(path, cv2.IMREAD_COLOR) # reading img
    cv2.imshow('Original', img1)  #display original img
    temp1 = np.array(img1) #converting to numpy array
    final=[]
    if flag == True:  #condition for mode(horizontal or vertical)
        temp3=[]
        temp4=[]
        for i in range(0,img1.shape[0]):  #iterate at rows of img
            temp2=img1.shape[1]-1
            temp4.clear()
            for j in range(0,img1.shape[1]):  #iterate at cols of img
                temp4.append(temp1[i][temp2])  #reading from end to starting index
                temp2=temp2-1
            temp3=np.array(temp4) #converting to numpy array
            final.append(temp3)

        img3=np.array(final)  #converting to numpy array
        cv2.imshow('Horizontal', img3) #displaying final img
        cv2.waitKey()
        final.clear()
    else:
        temp3=[]
        temp4=[]
        for i in range(0,img1.shape[0]): #shift 1
            temp2=img1.shape[1]-1
```



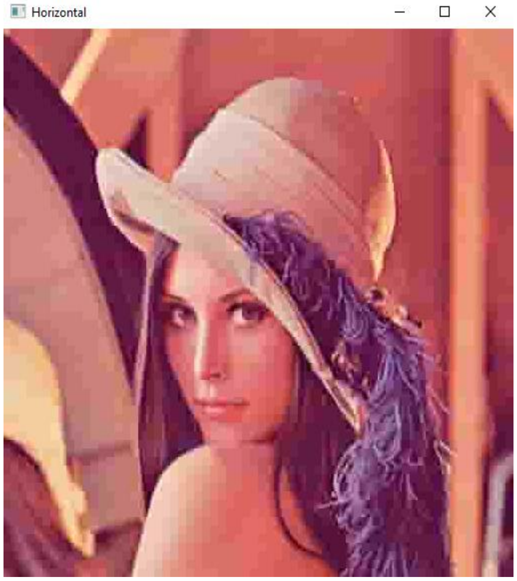
```
            temp4.clear()
            for j in range(0,img1.shape[1]):
                temp4.append(temp1[temp2][i])
                temp2=temp2-1
            temp3=np.array(temp4)
            final.append(temp3)
        img3=np.array(final) #converting to numpy array
        temp1 = np.array(img3) #converting to numpy array
        temp5=[]
        temp6=[]
        final1=[]
        for i in range(0,img3.shape[0]): #shift 2
            temp2=img3.shape[1]-1
            temp6.clear()
            for j in range(0,img3.shape[1]):
                temp6.append(temp1[temp2][i])
                temp2=temp2-1
            temp5=np.array(temp6) #converting to numpy array
            final1.append(temp5)
        img4=np.array(final1) #converting to numpy array

        cv2.imshow('Virtical', img4) #display img
        cv2.waitKey()
```

Input images:

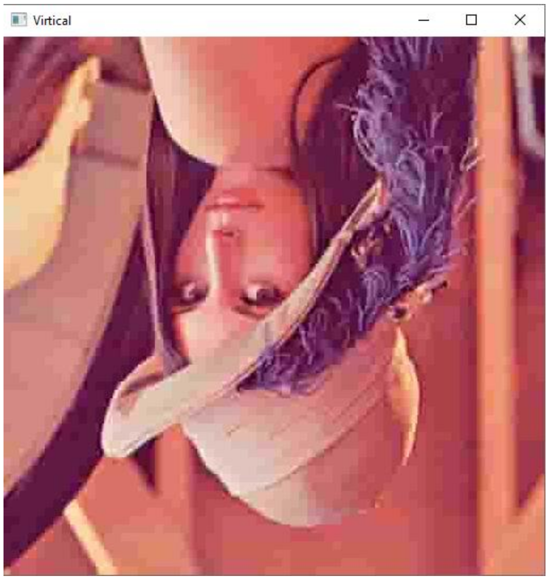


Output image:

Horizontal:



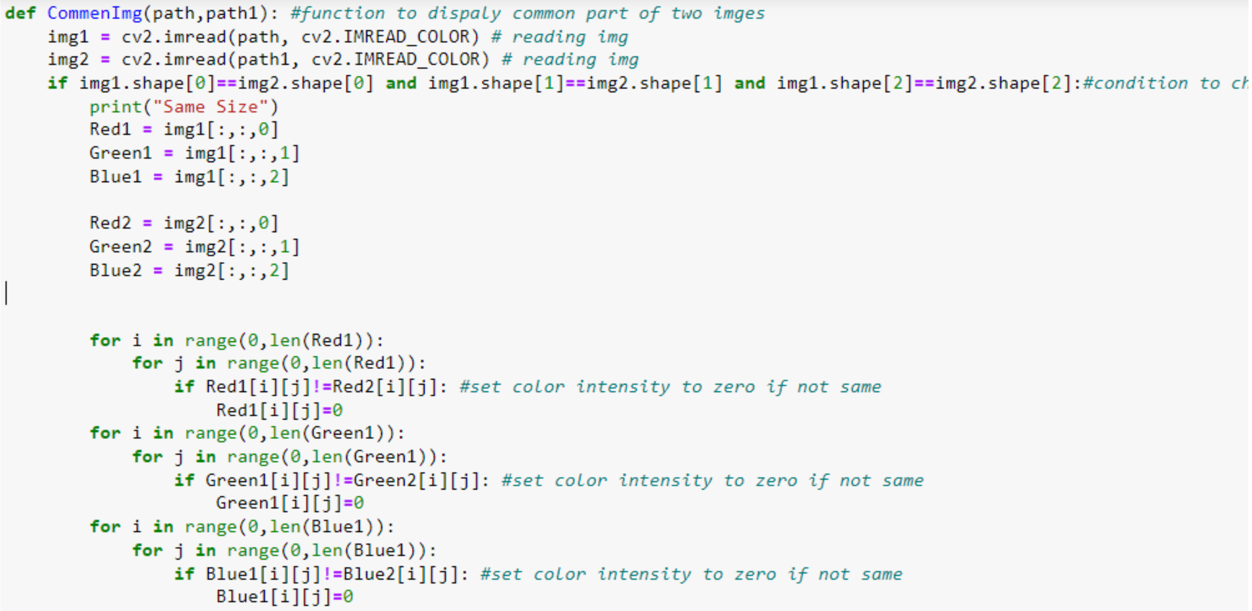
Vertical:



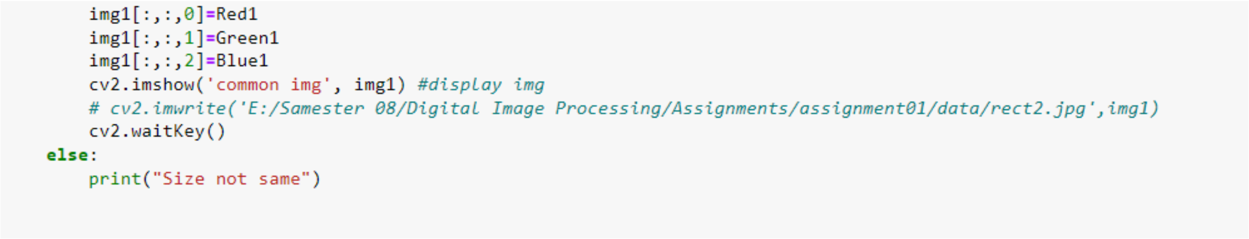
Question 05:

CommenImg(path,path1) function takes two images as input and finds the common part, and then displays it.

Code:

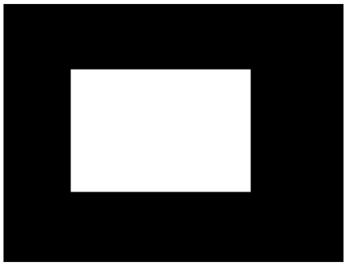


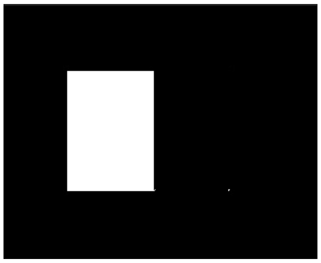
```
def CommenImg(path,path1): #function to dispaly common part of two imges
    img1 = cv2.imread(path, cv2.IMREAD_COLOR) # reading img
    img2 = cv2.imread(path1, cv2.IMREAD_COLOR) # reading img
    if img1.shape[0]==img2.shape[0] and img1.shape[1]==img2.shape[1] and img1.shape[2]==img2.shape[2]:#condition to ch
        print("Same Size")
        Red1 = img1[:,:,0]
        Green1 = img1[:,:,1]
        Blue1 = img1[:,:,2]

        Red2 = img2[:,:,0]
        Green2 = img2[:,:,1]
        Blue2 = img2[:,:,2]


        for i in range(0,len(Red1)):
            for j in range(0,len(Red1)):
                if Red1[i][j]!=Red2[i][j]: #set color intensity to zero if not same
                    Red1[i][j]=0
        for i in range(0,len(Green1)):
            for j in range(0,len(Green1)):
                if Green1[i][j]!=Green2[i][j]: #set color intensity to zero if not same
                    Green1[i][j]=0
        for i in range(0,len(Blue1)):
            for j in range(0,len(Blue1)):
                if Blue1[i][j]!=Blue2[i][j]: #set color intensity to zero if not same
                    Blue1[i][j]=0
```



```
        img1[:,:,0]=Red1
        img1[:,:,1]=Green1
        img1[:,:,2]=Blue1
        cv2.imshow('common img', img1) #display img
        # cv2.imwrite('E:/Samester 08/Digital Image Processing/Assignments/assignment01/data/rect2.jpg',img1)
        cv2.waitKey()
    else:
        print("Size not same")
```

Input images:





Output image: