1)Python program using OpenCV library and perform the following with respect to an image.

a) Read a gray scale image and display.
b) Increase and decrease the brightness and save it in a drive.
c) Split the image into three channel(R,G,B).
d) Resize the image by shrinking and zooming the same by using interpolation method.
e) Rotate the image to about 60 degree without scaling.
f) Demonstrate edge detection of your image with suitable algorithm.

RegNo:2117085
Date: 10-08-2022

---

a)

```
import cv2
img = cv2.imread(r"D:\IP\images\dog.jpg",cv2.IMREAD_GRAYSCALE)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

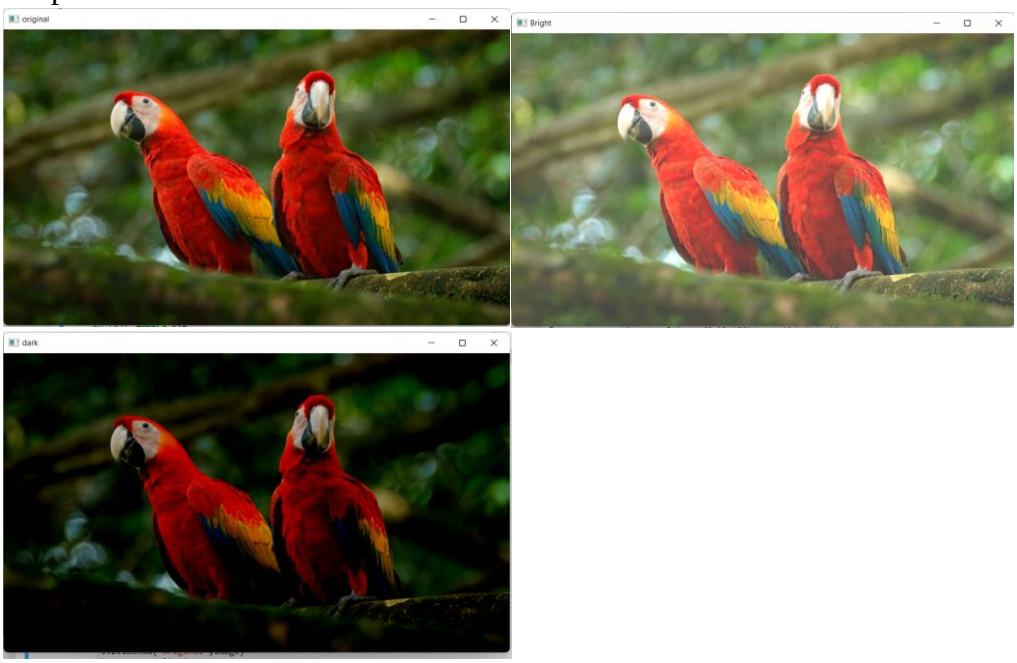
Output:

b)

```
import cv2
import numpy as np
path = r'D:\IP\images\bird.jpg'
image=cv2.imread(r'D:\IP\images\bird.jpg')
cv2.imshow("original",image)
Intensity_Matrix=np.ones(image.shape,dtype="uint8")*60
brt_image=cv2.add(image,Intensity_Matrix)
cv2.imshow("Bright",brt_image)
dark_image=cv2.subtract(image,Intensity_Matrix)
cv2.imshow("dark",dark_image)
cv2.waitKey(0)
```

print(Intensity_Matrix)

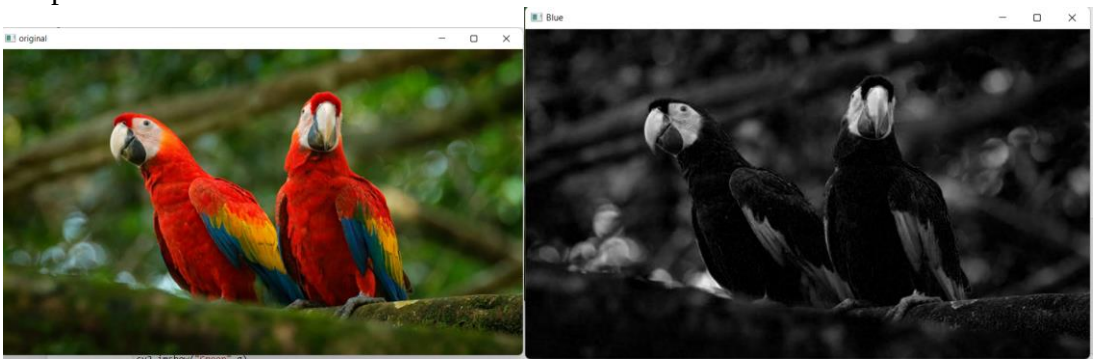
Output:

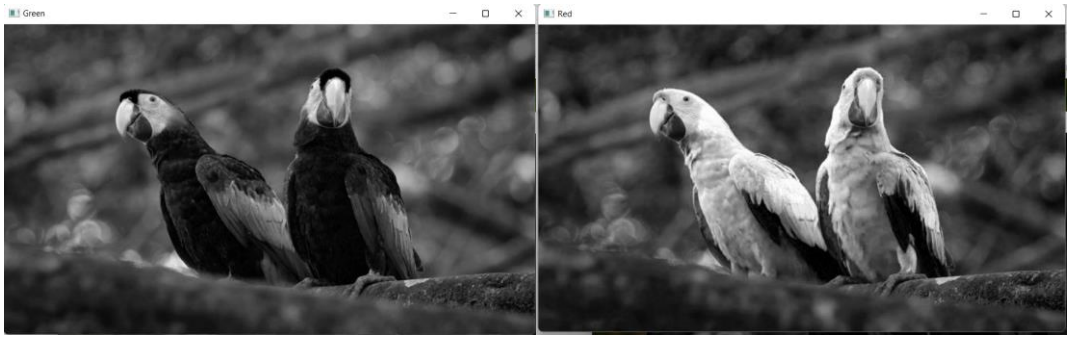


c)

```
import cv2
path = r'D:\IP\images\bird.jpg'
image=cv2.imread(r'D:\IP\images\bird.jpg')
cv2.imshow("original",image)
b,g,r=cv2.split(image)
cv2.imshow("Blue",b)
cv2.imshow("Green",g)
cv2.imshow("Red",r)
cv2.waitKey(0)
```

Output:

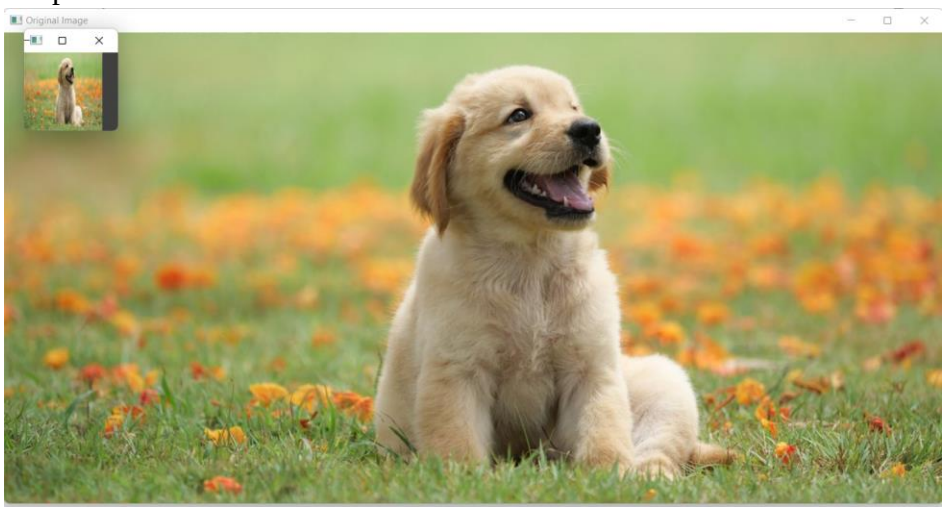




d)

```
import cv2
import numpy as np
image = cv2.imread(r"D:\IP\images\dog.jpg")
cv2.imshow('Original Image',image)
down_width=100
down_height=100
down_points=(down_width , down_height)
resized_down=cv2.resize(image,down_points,interpolation=cv2.INTER_LINEAR)
up_width=200
up_height=100
up_points=(up_width , up_height)
resized_up=cv2.resize(image,up_points,interpolation=cv2.INTER_LINEAR)
cv2.imshow('Resized Down by defining height and width',resized_down)
cv2.waitKey()
cv2.imshow('Resized Up by defining height and width',resized_down)
cv2.waitKey()
cv2.destroyAllWindows()
```

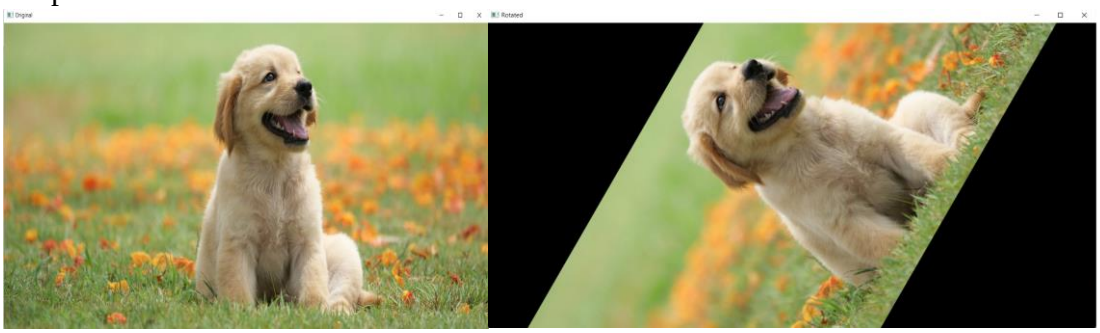
Output:



e)

```
import cv2
import imutils
path = r'D:\IP\images\dog.jpg'
img=cv2.imread(r"D:\IP\images\dog.jpg",cv2.IMREAD_COLOR)
rotate_img=imutils.rotate(img,angle=60)
cv2.imshow("Original",img)
cv2.imshow("Rotated",rotate_img)
cv2.waitKey(0)
```

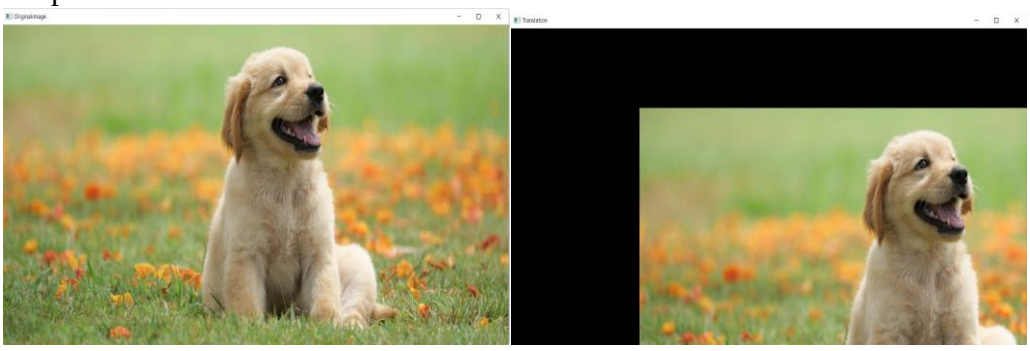
Output:



f)

```
import cv2
import numpy as np
path = r'D:\IP\images\dog.jpg'
img=cv2.imread(r"D:\IP\images\dog.jpg",cv2.IMREAD_COLOR)
height, width=img.shape[:2]
quater_height,quater_width = height/4, width/4
T= np.float32([[1,0,quater_width],[0,1,quater_height]])
img_translation=cv2.warpAffine(img,T,(width,height))
cv2.imshow("Originalimage",img)
cv2.imshow('Translation',img_translation)
cv2.waitKey()
cv2.destroyAllWindows()
```

Output:

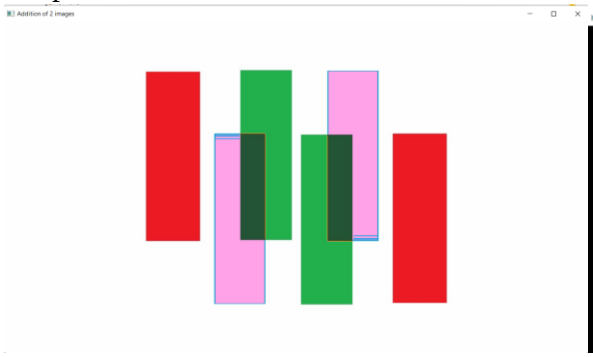


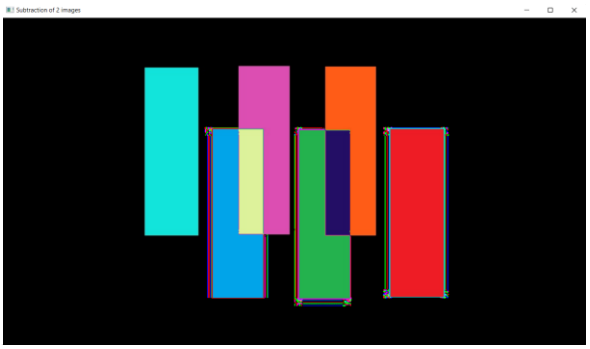
2)Write a python program using open CV library and perform basic image processing operations(Arithmetic and logical).
RegNo:2117085
Date:10-08-2022

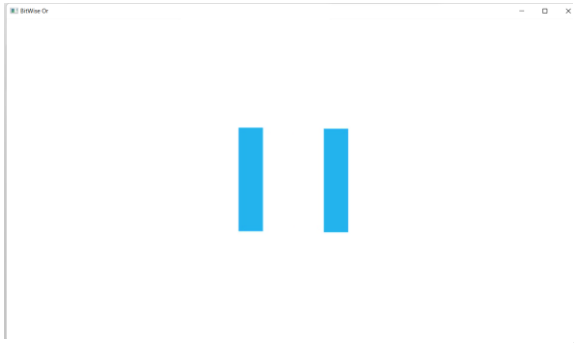
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
path1 = r'D:\IP\images\RGB.jpg'
path2 = r'D:\IP\images\RGB2.jpg'
image1=cv2.imread(path1)
image2=cv2.imread(path2)
addImage = np.add(image1,image2)

cv2.imshow("Addition of 2 images",addImage)
subImage = np.subtract(image1,image2)
cv2.imshow("Subtraction of 2 images",subImage)
bitOr = cv2.bitwise_or(image1,image2,mask=None)
cv2.imshow("BitWise Or",bitOr)
bitAnd = cv2.bitwise_and(image1,image2,mask=None)
cv2.imshow("BitWise And",bitAnd)
bitXor = cv2.bitwise_xor(image1,image2,mask=None)
cv2.imshow("BitWise Xor",bitXor)
bitNot = cv2.bitwise_not(image1)
cv2.imshow("BitWise not",bitNot)
cv2.waitKey(0)
```

Original image:

Output:


Addition of 2 images


Subtraction of 2 images


BitWise Or


BitWise And


BitWise not


BitWise Xor