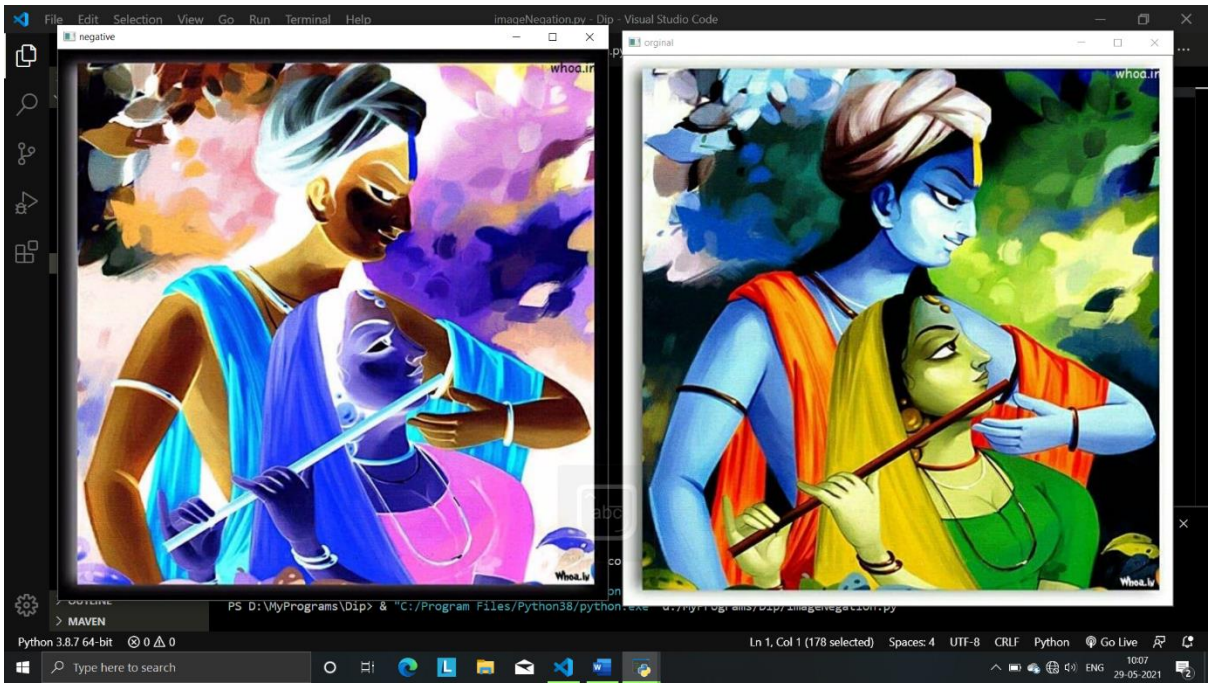
PROGRAM 1

Image Negative

```
import cv2
img=cv2.imread("image\My.jpg")
imgNeg=255-img
cv2.imshow("orginal",img)
cv2.imshow("negative",imgNeg)
cv2.imwrite("image\MyNegative.jpg",imgNeg)
cv2.waitKey(0)
```

OUTPUT

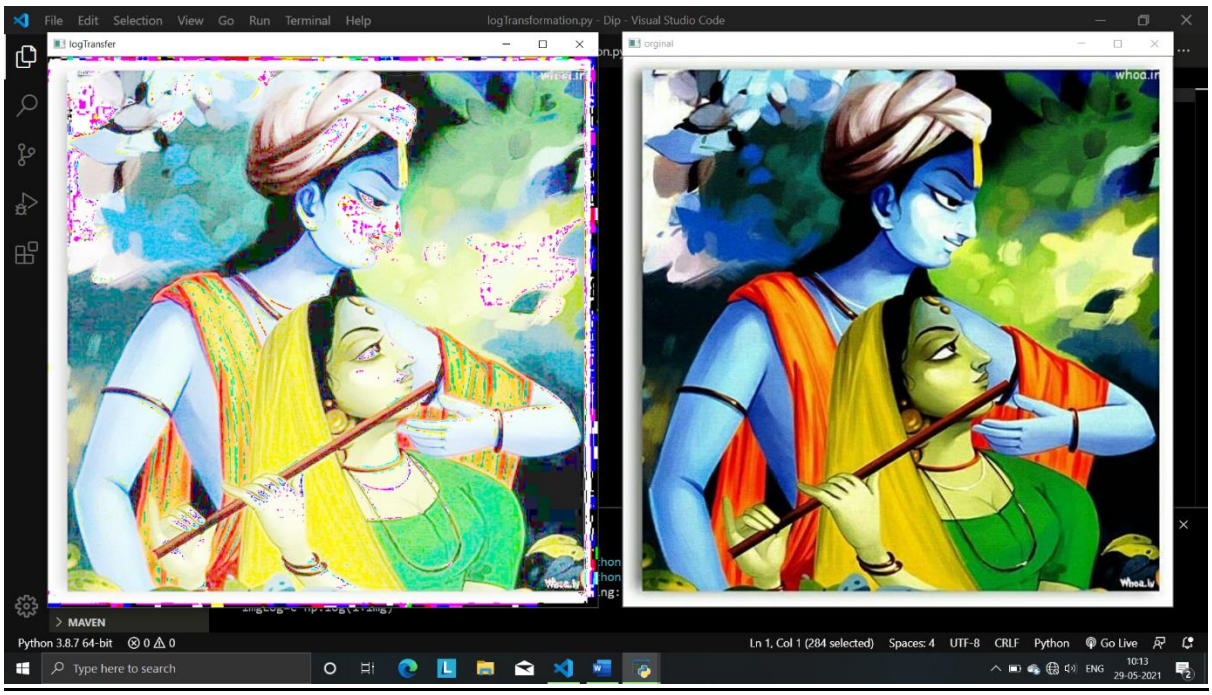


PROGRAM 2

Log Transformation

```
import cv2
import numpy as np
img=cv2.imread('image\My.jpg')
c=256/(np.log(1+np.max(img)))
imgLog=c*np.log(1+img)
imgLog=np.array(imgLog,dtype=np.uint8)
cv2.imshow("orginal",img)
cv2.imshow("logTransfer",imgLog)
cv2.imwrite("image\MyLogTranfered.jpg",imgLog)
cv2.waitKey(0)
```

OUTPUT

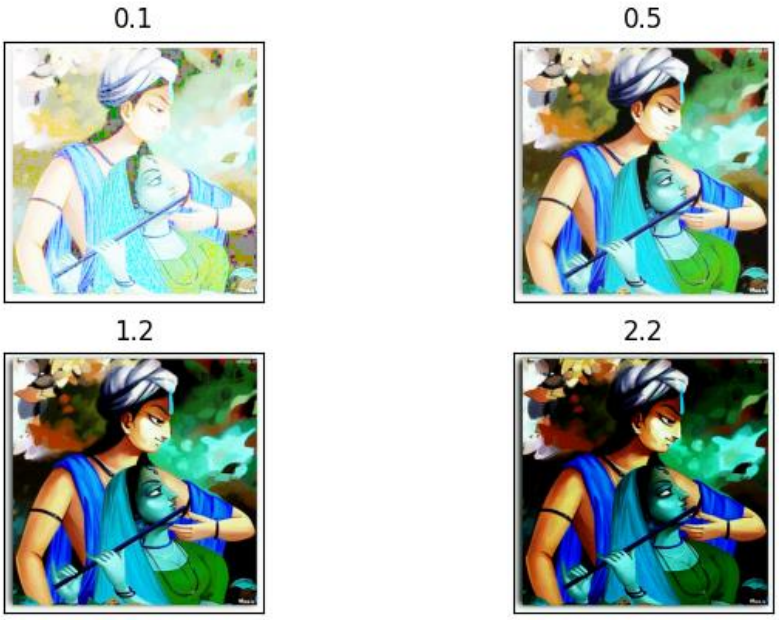


PROGRAM 3

Power-low Transformation

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
img=cv2.imread("image\My.jpg")
i=0
for gamma in [0.1,0.5,1.2,2.2]:
    gammaImg=np.array(255*(img/255)**gamma,dtype="uint8")
    cv2.imwrite("image\MyGamma"+str(gamma)+".jpg",gammaImg)
    i=i+1
    plt.subplot(2,2,i),plt.imshow(gammaImg,cmap="gray"),plt.title(str(gamma)),plt.xticks([]),plt.yticks([])
plt.show()
cv2.waitKey(0)
```

OUTPUT

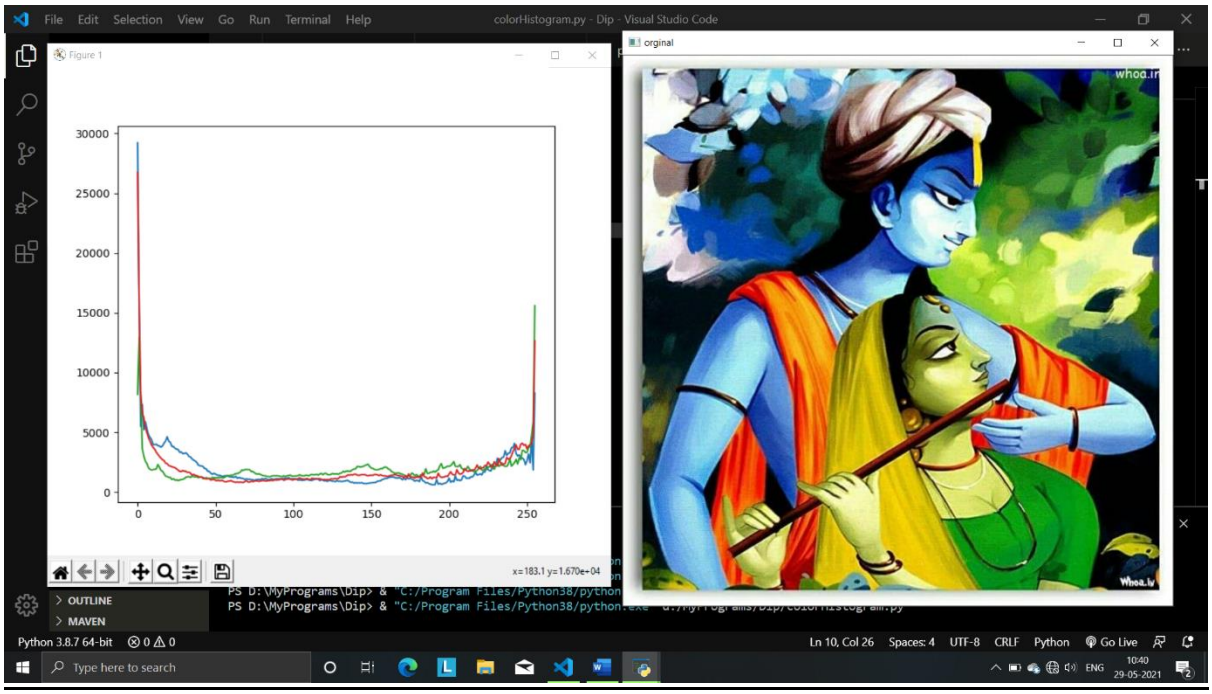


PROGRAM 4

Histogram of a image

```
from matplotlib import pyplot as plt
import cv2
img=cv2.imread("image\My.jpg",1)
blue=cv2.calcHist([img],[0],None,[256],[0,256])
green=cv2.calcHist([img],[1],None,[256],[0,256])
red=cv2.calcHist([img],[2],None,[256],[0,256])
plt.plot(blue,color='tab:blue')
plt.plot(green,color='tab:green')
plt.plot(red,color='tab:red')
cv2.imshow("orginal",img)
plt.show()
cv2.waitKey(0)
```

OUTPUT

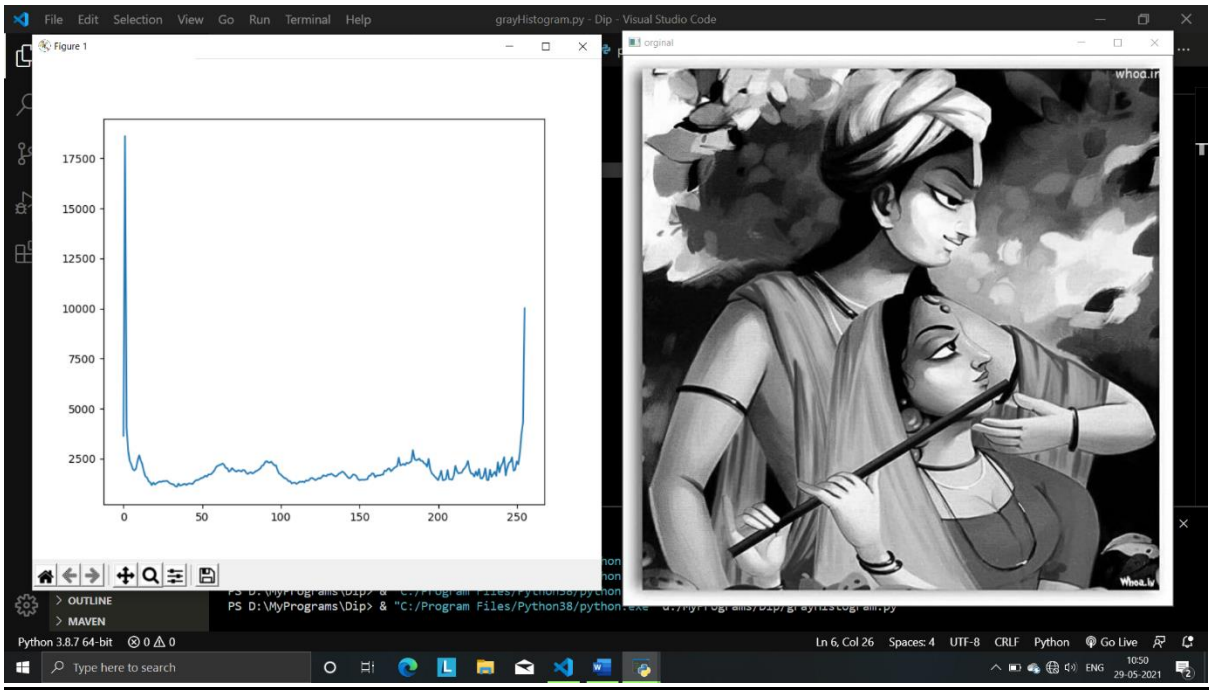


PROGRAM 5

Histogram of a GrayScale Image

```
from matplotlib import pyplot as plt
import cv2
img=cv2.imread("image\My.jpg",0)
hstr=cv2.calcHist([img],[0],None,[256],[0,256])
plt.plot(hstr)
cv2.imshow("orginal",img)
plt.show()
cv2.waitKey(0)
```

OUTPUT

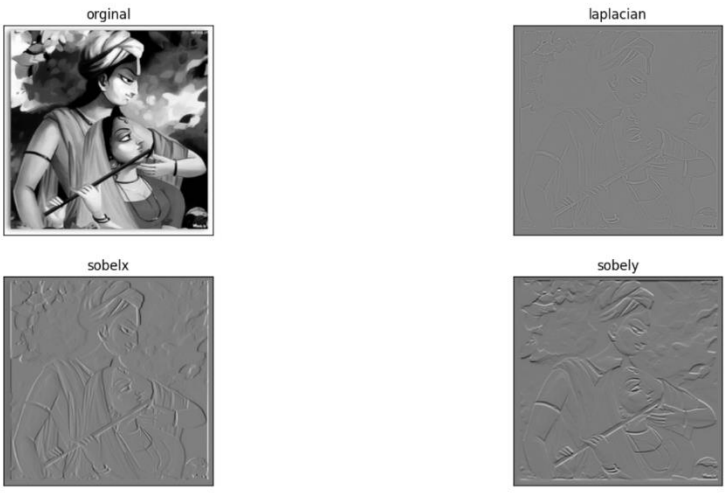


PROGRAM 6

Edge Detection

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img0=cv2.imread("image\My.jpg",)
gray=cv2.cvtColor(img0,cv2.COLOR_BGR2GRAY)
img=cv2.GaussianBlur(gray,(3,3),0)
laplacian=cv2.Laplacian(img,cv2.CV_64F)
sobelx=cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)
sobely=cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)
plt.subplot(2,2,1),plt.imshow(img,cmap="gray"),plt.title("orginal"),plt.xticks([]),plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap="gray"),plt.title("laplacian"),plt.xticks([]),plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobelx,cmap="gray"),plt.title("sobelx"),plt.xticks([]),plt.yticks([])
plt.subplot(2,2,4),plt.imshow(sobely,cmap="gray"),plt.title("sobely"),plt.xticks([]),plt.yticks([])
plt.show()
```

OUTPUT

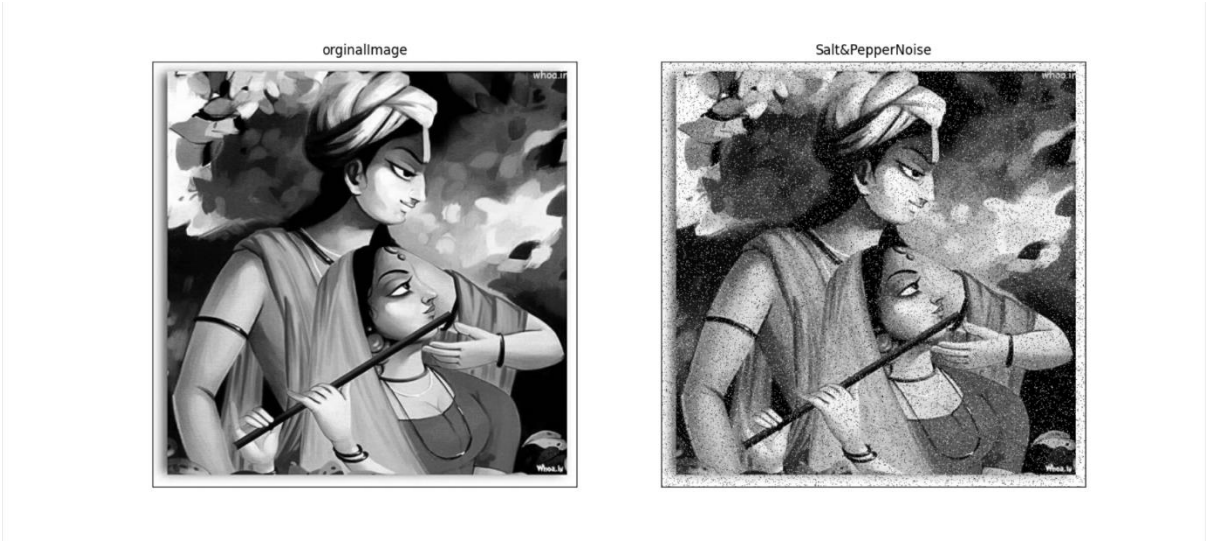


PROGRAM 7

Salt and pepper Noice

```
import cv2
import numpy as np
import random
from matplotlib import pyplot as plt
def sp_noice(image,prob):
   output=np.zeros(image.shape,np.uint8)
   thres=1-prob
   for i in range(image.shape[0]):
      for j in range(image.shape[1]):
         rdn=random.random()
         if(rdn<prob):
            output[i][j]=0
         elif(rdn>thres):
            output[i][j]=255
         else:
            output[i][j]=image[i][j]
   return output
img=cv2.imread("image\My.jpg",0)
blur=sp_noice(img,0.05)
plt.subplot(121),plt.imshow(img,cmap='gray'),plt.xticks([]),plt.yticks([]),plt.title("orginalImage")
plt.subplot(122),plt.imshow(blur,cmap='gray'),plt.xticks([]),plt.yticks([]),plt.title("Salt&PepperNoise")
cv2.imwrite("image\MySalt&Pepper.jpg",blur)
plt.show()
```

OUTPUT



PROGRAM 8

Meadian Filtering

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img=cv2.imread("image\MySalt&Pepper.jpg",0)
m,n=img.shape
imgnew=np.zeros([m,n])
for i in range(1,m-1):
   for j in range(1,n-1):
      tmp=[img[i-1,j-1],img[i-1,j],img[i-1,j+1],img[i,j-1],img[i,j],img[i,j+1],img[i+1,j-1],img[i+1,j],img[i+1,j+1]]
      tmp=sorted(tmp)
      imgnew[i,j]=tmp[4]
imgNew=imgnew.astype(np.uint8)
plt.subplot(121),plt.imshow(img,cmap='gray'),plt.xticks([]),plt.yticks([]),plt.title("salt&peperNoice")
plt.subplot(122),plt.imshow(imgnew,cmap='gray'),plt.xticks([]),plt.yticks([]),plt.title("filteredImage")
cv2.imwrite("image\MyMedianFiltered.jpg",imgnew)
plt.show()
print("succes")
cv2.waitKey()
```

OUTPUT