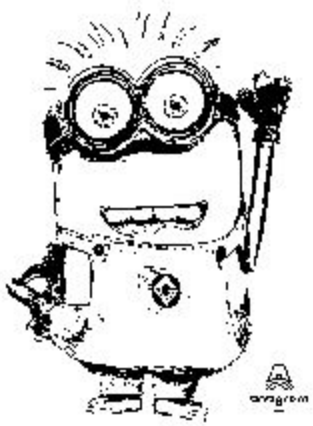
EXP 7: Implementation of Contrast stretching, Dynamic range compression and bit plane slicing.

#Bit scaling:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

def decimalToBinary(n):
    temp = bin(n).replace("0b","")
    rem = len(temp) % 8
    if(rem != 0):
        append = "0"* (8-rem)
        temp = append + temp
    return(temp)

img = cv.imread("minion.jpg",0)
row, col = img.shape

cv.imshow("Original Image",img)

output = np.array(np.zeros((row,col), np.float32))
r = 6 #int(input("Enter value: "))

for i in range (row):
    for j in range (col):
        c = decimalToBinary(img[i][j])
        d = c[r]
        output[i][j] = int(d)
        if(output[i][j] == 1):
            output[i][j] = 255
output = output.astype('uint8')

cv.imshow("Binary Slice",output)

cv.waitKey(0)
cv.destroyAllWindows()
```

#OUTPUT:



#Contrast stretching:

```
import cv2 as cv
from google.colab.patches import cv2_imshow
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("minion.jpg",0)
row, col = img.shape
img_temp = img.astype('float32')

output = np.array(np.zeros((row,col), np.uint8))
LT = 65
#int(input("Enter lower threshold value: "))
UT = 127
#int(input("Enter higher threshold value: "))

for i in range (row):
    for j in range (col):
        if(img[i][j] <= LT):
            output[i][j] = img[i][j] // 2
        elif(img[i][j] <= UT):
            output[i][j] = 2 * (img[i][j] -LT) + (LT // 2)
        else:
            output[i][j] = (img[i][j] - UT) // 2  + (LT // 2) + (2 *(UT - LT))

cv2_imshow(img)
cv2_imshow(output)

cv.waitKey(0)
cv.destroyAllWindows()
```

#OUTPUT:

#Dynamic Range:

```
import cv2 as cv
from google.colab.patches import cv2_imshow
import numpy as np
import math

from matplotlib import pyplot as plt

aa = cv.imread("minion.jpg",0)
a = aa.astype('float32')

row, col = aa.shape

c = np.array(np.zeros((row,col), np.float32))

for i in range (row):
    for j in range (col):
        c[i][j] = a[i][j] * ((-1)**(i+j))

f = np.fft.fft2(c)
ft = np.fft.fftshift(f)
d = np.abs(f)
d_log = (np.log(d+1)/(np.log(1+np.max(d))))*255
d_log = np.array(d_log,dtype=np.uint8)
cv2_imshow(aa)

cv2_imshow(d_log )

cv2_imshow(d.astype('uint8') )

cv.waitKey(0)
cv.destroyAllWindows()
```

#OUTPUT:

EXP 6: Implementation of Image Negative, Gray level slicing and Thresholding

#Negative:

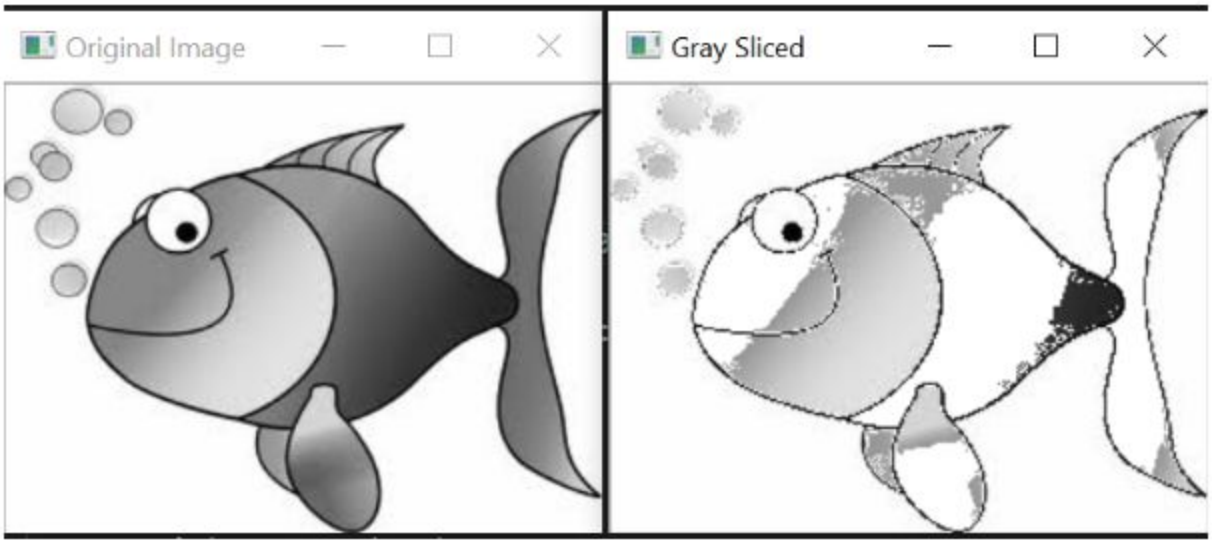
```
import cv2 as cv
from google.colab.patches import cv2_imshow
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("minion.jpg",0)
row, col = img.shape
output = np.array(np.zeros((row,col), np.uint8))

cv2_imshow(img)

for i in range (row):
    for j in range (col):
        img[i][j] = 255 - img[i][j]

cv2_imshow(img)

cv.waitKey(0)
cv.destroyAllWindows()
```

#OUTPUT:

#Gray level slicing:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("minion.jpg",0)
row, col = img.shape
output = np.array(np.zeros((row,col), np.uint8))

cv.imshow("Original Image",img)

for i in range (row):
    for j in range (col):
        if(img[i][j] > 50 and img[i][j] < 150):
            img[i][j] = 255
        else:
            img[i][j] = img[i][j]

cv.imshow("Gray Sliced",img)

cv.waitKey(0)
cv.destroyAllWindows()
```

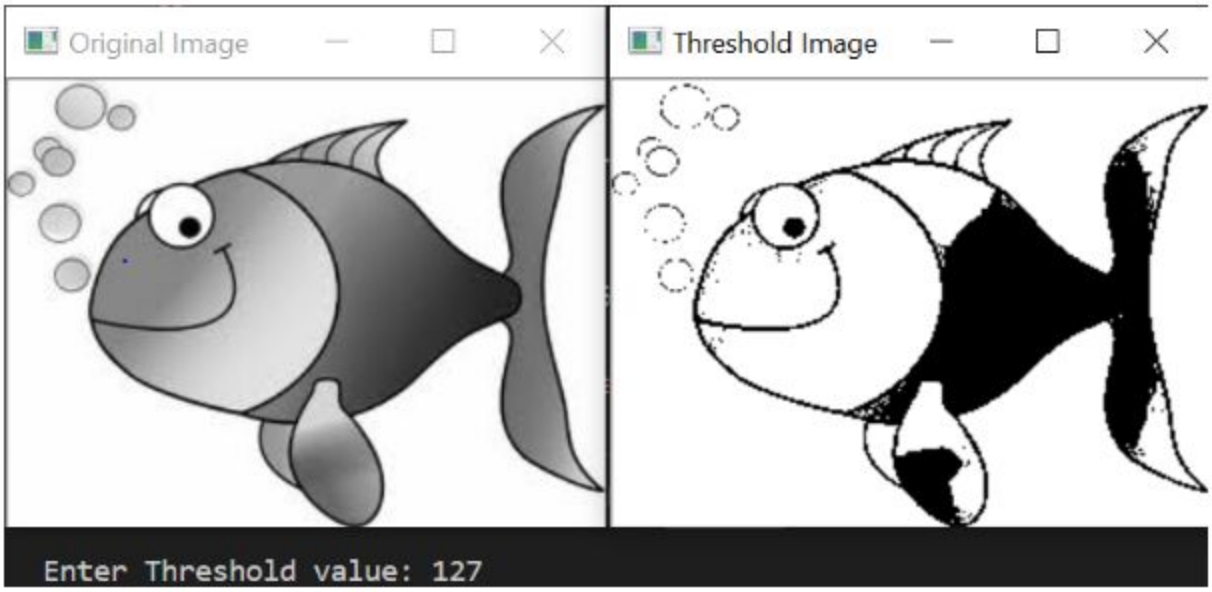
#OUTPUT:



#Thresholding:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("minion.jpg",0)
row, col = img.shape
output = np.array(np.zeros((row,col), np.uint8))

thresh = int(input("Enter Threshold value: "))
cv.imshow("Original Image",img)

for i in range (row):
    for j in range (col):
        if(img[i][j] < thresh):
            img[i][j] = 0
        else:
            img[i][j] = 255

cv.imshow("Threshold Image",img)

cv.waitKey(0)
cv.destroyAllWindows()
```

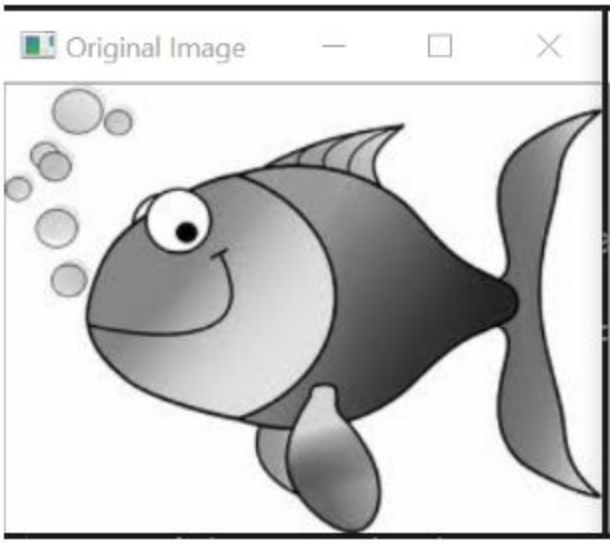
#OUTPUT:

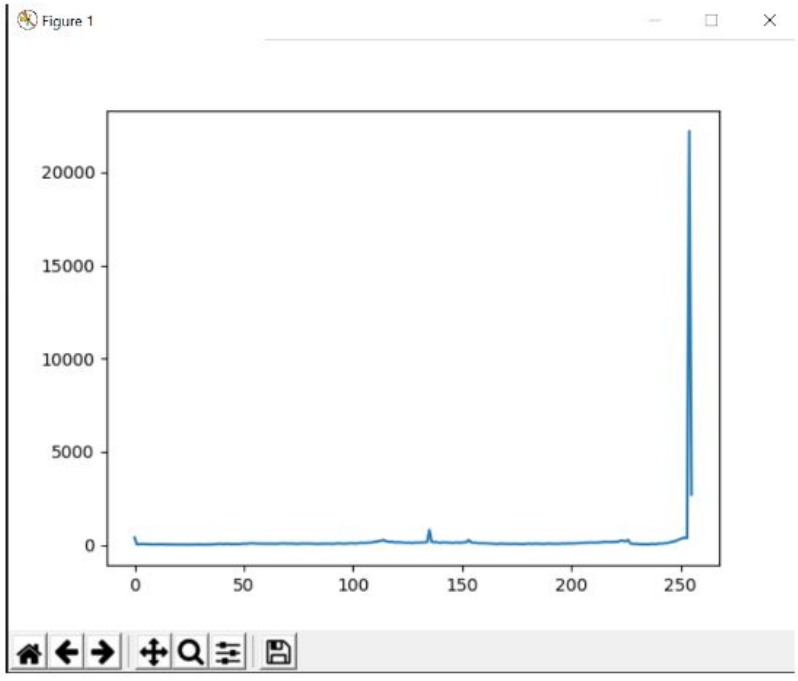


EXP 8: Implementation of Histogram processing

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img=cv2.imread("minion.jpg",0)
row,col=img.shape
img1=np.array(np.zeros((row,col),np.uint8))
for x in range(row):
  for y in range(col):
    if img[x][y]==0:
      img[x][y]=0
histr=cv2.calcHist([img],[0],None,[256],[0,256])
plt.plot(histr)
plt.show()
```

#OUTPUT:

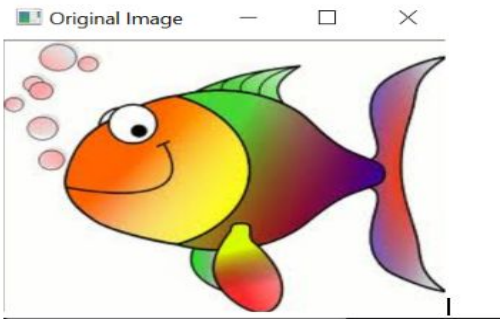


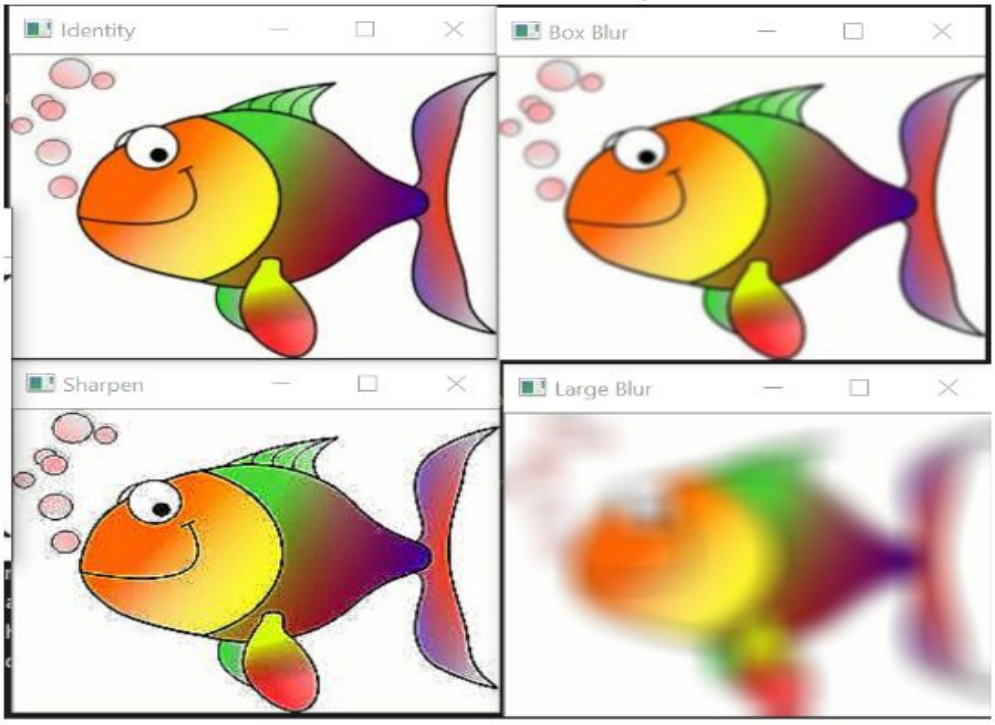


EXP 9: Implementation of Image Smoothing/Image Sharpening.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread("minion.jpg",1) # 0 for bw and 1 for color
impulse = np.array(([0,0,0],[0,1,0],[0,0,0]), np.float32) #identity
largeBlur = np.ones((21, 21), dtype="float") * (1.0 / (21 * 21))
sharpen = np.array(([0,-1,0],[-1,5,-1],[0,-1,0]), np.float32) #sharpen
boxblur = np.array(np.ones((3,3), np.float32))/9 #box blur
output1 = cv.filter2D(img, -1, impulse)
output6 = cv.filter2D(img, -1, largeBlur)
output7 = cv.filter2D(img, -1, sharpen)
output8 = cv.filter2D(img, -1, boxblur)
cv.imshow("Original Image",img)
cv.imshow("Identity",output1)
cv.imshow("Large Blur",output6)
cv.imshow("Sharpen",output7)
cv.imshow("Box Blur",output8)
cv.waitKey(0)
cv.destroyAllWindows()
```

#OUTPUT:





EXP 10: Implementation of Edge detection using Sobel and Previtt masks.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread("minion.jpg",1) # 0 for bw and 1 for color

sobely = np.array(([-1,-2,-1],[0,0,0],[1,2,1]), np.float32) #edge
detection1
sobelx = np.array(([-1,0,1],[-2,0,2],[-1,0,1]), np.float32) #edge
detection1
previtty = np.array(([-1,-1,-1],[0,0,0],[1,1,1]), np.float32) #edge
detection2
previttx = np.array(([-1,0,1],[-1,0,1],[-1,0,1]), np.float32) #edge
detection2

output2 = cv.filter2D(img, -1, sobely)
output3 = cv.filter2D(img, -1, sobelx)
output4 = cv.filter2D(img, -1, previtty)
output5 = cv.filter2D(img, -1, previttx)

cv.imshow("Original Image",img)
cv.imshow("Sobel-y",output2)
cv.imshow("Sobel-x",output3)
cv.imshow("Previtt-y",output4)
cv.imshow("Previtt-x",output5)

cv.waitKey(0)
cv.destroyAllWindows()
```

#OUTPUT: