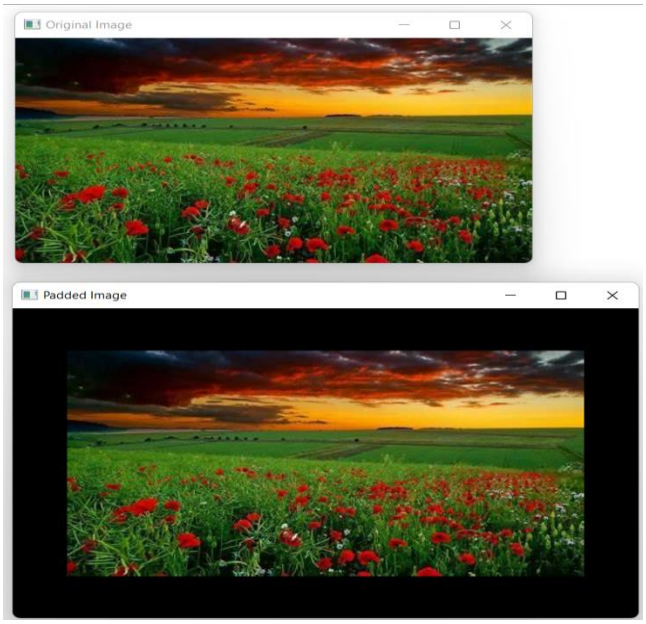
MAJOR PROJECT 2:

Create any of the Image Processing Projects using Numpy and/or OpenCV.(Projects done in the class are not accepted)
(One can use the haarcasacde models if necessary)

#1.PADDLING BLACK SPACES

```
import cv2
# Load the image
img = cv2.imread("abc.jpg")
# Get the shape of the image
h, w = img.shape[:2]
# Define the padding values for top, bottom, left and right
top, bottom, left, right = (50, 50, 50, 50)
#image Create a black image with the same size as the original
color = [0, 0, 0]
padded_img = cv2.copyMakeBorder(img, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color)
# Show the original and padded image
cv2.imshow("Original Image", img)
cv2.imshow("Padded Image", padded_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:



#2.COLOR REDUCTION

```
import cv2
import numpy as np
# Load the image
img = cv2.imread("abc.jpg")
# Create a color quantization object
cluster_number = 64
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
z = img.reshape((-1,3))
z = np.float32(z)
# Perform K-means clustering
ret,label,center=cv2.kmeans(z,cluster_number,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
# Compute the quantized image
center = np.uint8(center)
res = center[label.flatten()]
reduced_color_img = res.reshape((img.shape))
# Show the original and quantized image
cv2.imshow("Original Image", img)
cv2.imshow("Quantized Image", reduced_color_img)
cv2.waitKey(0)
```

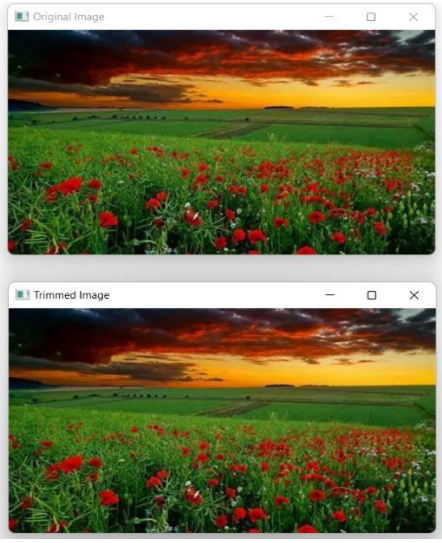
```
cv2.destroyAllWindows()
```

OUTPUT:

#3.TRIM IMAGE

```
import cv2
# Load the image
img = cv2.imread("abc.jpg")
# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find the coordinates of all non-black pixels
coords = cv2.findNonZero(gray)
# Get the bounding box of those non-black pixels
x, y, w, h = cv2.boundingRect(coords)
# Get the region of interest (ROI)
roi = img[y:y+h, x:x+w]
# Show the original and trimmed image
cv2.imshow("Original Image", img)
cv2.imshow("Trimmed Image", roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

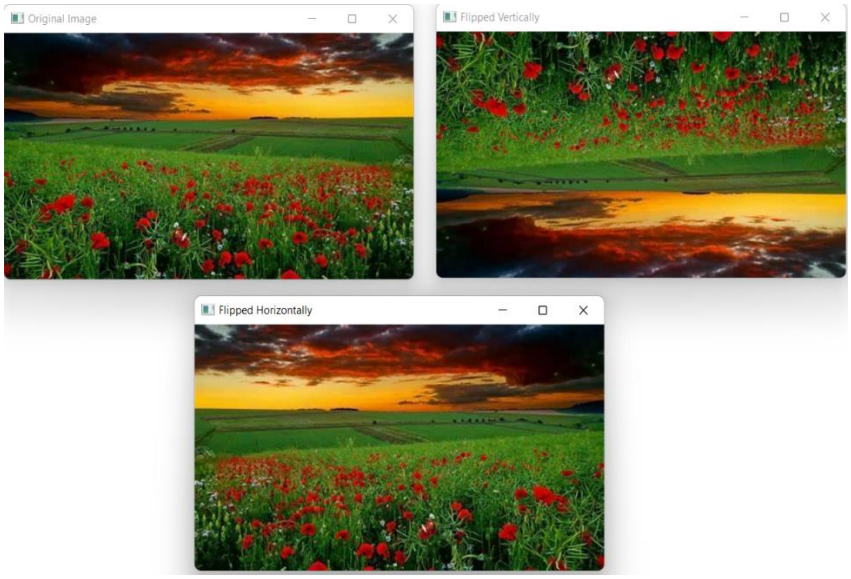
OUTPUT:



#4.FLIP IMAGE

```
import cv2
# Load the image
img = cv2.imread("abc.jpg")
# Flip the image horizontally
flip_horizontal = cv2.flip(img, 1)
# Flip the image vertically
flip_vertical = cv2.flip(img, 0)
# Show the original, flipped horizontally and flipped vertically image
cv2.imshow("Original Image", img)
cv2.imshow("Flipped Horizontally", flip_horizontal)
cv2.imshow("Flipped Vertically", flip_vertical)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

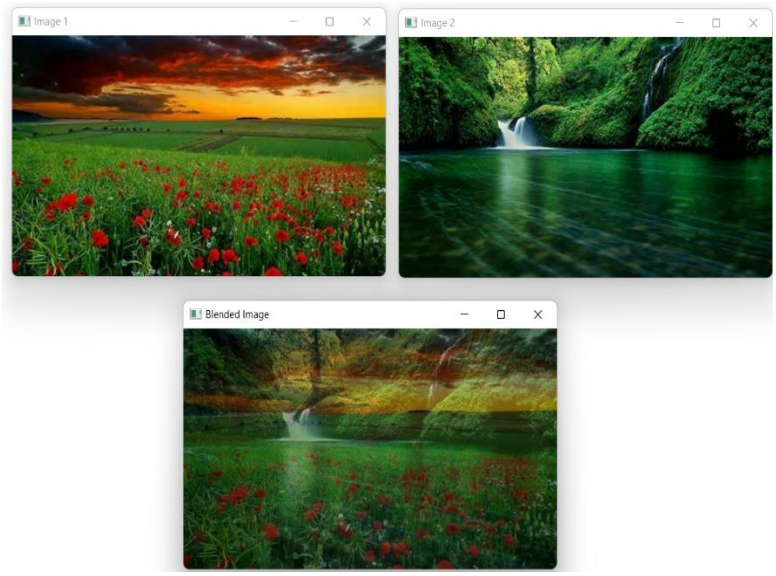
OUTPUT:



#5.BLENDING TWO IMAGES

```
import cv2
# Load the two images
img1 = cv2.imread("abc.jpg")
img2 = cv2.imread("123.jpg")
# Define the blending weight
alpha = 0.5  # Change this value to adjust the blending level
# Perform the blending
blended_img = cv2.addWeighted(img1, alpha, img2, 1 - alpha, 0)
# Show the original images and blended image
cv2.imshow("Image 1", img1)
cv2.imshow("Image 2", img2)
cv2.imshow("Blended Image", blended_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:



#6.MASKING IMAGES

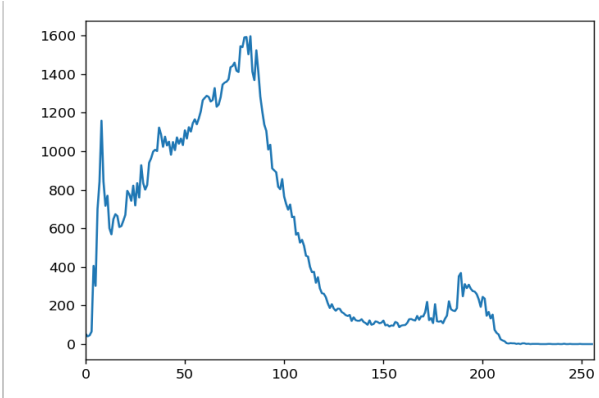
```
import cv2
# Load the image and the mask
img = cv2.imread("abc.jpg")
mask = cv2.imread("123.jpg", 0)
# Perform the masking
masked_img = cv2.bitwise_and(img, img, mask=mask)
# Show the original and masked image
cv2.imshow("Original Image", img)
cv2.imshow("Masked Image", masked_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

#7.HISTOGRAM FOR PIXEL INTENSITY

```
import cv2
import matplotlib.pyplot as plt
# Load the image
img = cv2.imread("abc.jpg", cv2.IMREAD_GRAYSCALE)
# Compute the histogram of pixel intensities
hist = cv2.calcHist([img], [0], None, [256], [0,256])
# Plot the histogram
plt.plot(hist)
plt.xlim([0,256])
plt.show()
```

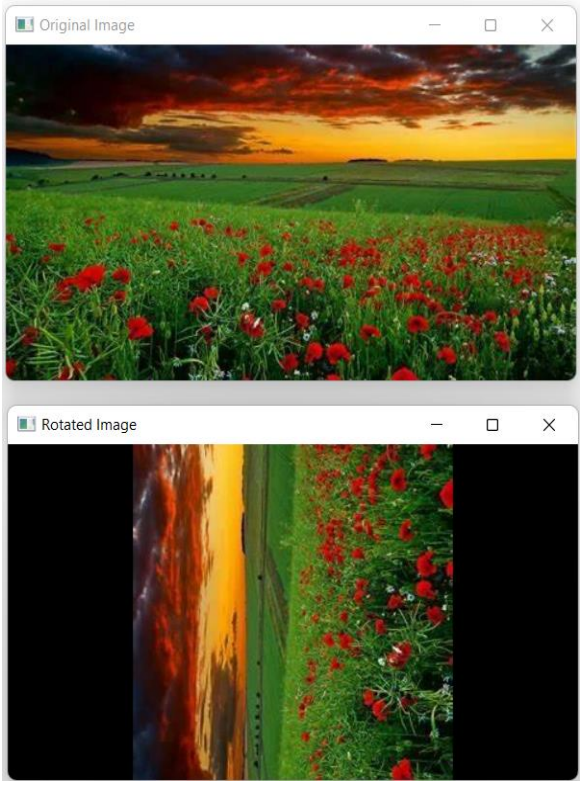
OUTPUT:



#8.ROTATING 90 DEGREES

```
import cv2

# Load the image

img = cv2.imread("abc.jpg")

# Get the image shape

rows, cols = img.shape[:2]

# Define the rotation matrix (clockwise)

M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)

# Perform the rotation (clockwise)

rotated_img = cv2.warpAffine(img, M, (cols,rows))

# Show the original and rotated image

cv2.imshow("Original Image", img)

cv2.imshow("Rotated Image", rotated_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



#9.PASTING WITH SLICE

```
import cv2

# Load the two images
```

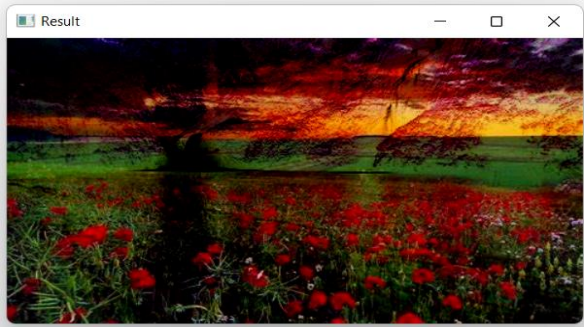
```
img1 = cv2.imread("abc.jpg")
img2 = cv2.imread("123.jpg")
# Define the slice of the second image to paste
x, y, w, h = (50, 50, 100, 100)
img2_slice = img2[y:y+h, x:x+w]
# Define the location to paste the slice
x, y = (100, 100)
# Paste the slice onto the first image
img1[y:y+h, x:x+w] = img2_slice
# Show the modified image
cv2.imshow("Modified Image", img1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

#10.SUBTRACTING IMAGES

```
import cv2
# Read the two images
img1 = cv2.imread('abc.jpg')
img2 = cv2.imread('123.jpg')
# Subtract the images
result = cv2.subtract(img1, img2)
# Show the result
cv2.imshow('Result', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

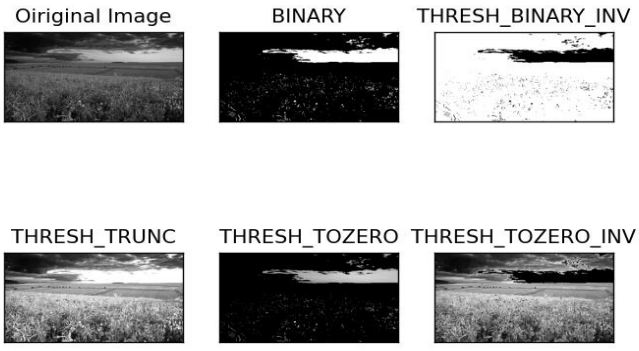
OUTPUT:



#11.SIMPLE THRESHOLD

```
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
#here 0 means that the image is loaded in gray scale format
gray_image = cv2.imread('abc.jpg',0)
ret,thresh_binary = cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY)
ret,thresh_binary_inv=cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY_INV)
ret,thresh_trunc=cv2.threshold(gray_image,127,255,cv2.THRESH_TRUNC)
ret,thresh_tozero=cv2.threshold(gray_image,127,255,cv2.THRESH_TOZERO)
ret,thresh_tozero_inv=cv2.threshold(gray_image,127,255,cv2.THRESH_TOZERO_INV)
#DISPLAYING THE DIFFERENT THRESHOLDING STYLES
names=['Oiriginal
Image','BINARY','THRESH_BINARY_INV','THRESH_TRUNC','THRESH_TOZERO','THRESH_TOZERO_INV']
images=gray_image,thresh_binary,thresh_binary_inv,thresh_trunc,thresh_tozero,thresh_tozero_inv
for i in range(6):
        plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
        plt.title(names[i])
        plt.xticks([]),plt.yticks([])
plt.show()
```

OUTPUT:

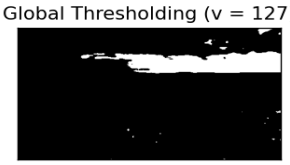


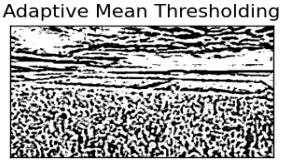
#12.ADAPTIVE THRESHOLD

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('abc.jpg',0)
img = cv2.medianBlur(img,5)
ret,th1=cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)
th3=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)',
'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]
for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

OUTPUT:


Original Image


Global Thresholding (v = 127)


Adaptive Mean Thresholding


Adaptive Gaussian Thresholding