```
#1_st program.
#Write a Program to read a digital image. Split and display image into 4
quadrants, up, down, right and left

#TRAINGLE FORMAT


import cv2

# Load the image
image = cv2.imread("CAR.jpg")

# Get the height and width of the image
height, width = image.shape[:2]



# Split the image into four quadrants
up_quadrant = image[0:height//2, 0:width//2]  # Upper left quadrant
right_quadrant = image[0:height//2, width//2:width]  # Upper right quadrant
down_quadrant = image[height//2:height, 0:width//2]  # Lower left quadrant
left_quadrant = image[height//2:height, width//2:width]  # Lower right quadrant

# Draw triangular format partition lines
cv2.line(image, (0, 0), (width//2, height//2), (0, 255, 0), thickness=2)  #
Diagonal line from top left to center
cv2.line(image, (width, 0), (width//2, height//2), (0, 255, 0), thickness=2)  #
Diagonal line from top right to center
cv2.line(image, (0, height), (width//2, height//2), (0, 255, 0), thickness=2)  #
Diagonal line from bottom left to center
cv2.line(image, (width, height), (width//2, height//2), (0, 255, 0),
thickness=2)  # Diagonal line from bottom right to center

# Combine the quadrants into one image
combined_image = cv2.vconcat([cv2.hconcat([up_quadrant, right_quadrant]),
cv2.hconcat([down_quadrant, left_quadrant])])

# Display the image with partitions
cv2.imshow("Image with Partitions", image)

# Display the combined image
cv2.imshow("Combined Image", combined_image)

# Wait for a key press and then close all windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
#program 2
#Write a program to showrotation, scaling, and translation of an image.

import cv2

# Reading the image
image = cv2.imread('CAR_bw.jpg')

# dividing height and width by 2 to get the center of the image
height, width = image.shape[:2]
# get the center coordinates of the image to create the 2D rotation matrix
center = (width/2, height/2)

# using cv2.getRotationMatrix2D() to get the rotation matrix
rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=180, scale=1)

# rotate the image using cv2.warpAffine
rotated_image = cv2.warpAffine(src=image, M=rotate_matrix, dsize=(width,
height))

cv2.imshow('Original image', image)
cv2.imshow('Rotated image', rotated_image)
# wait indefinitely, press any key on keyboard to exit
cv2.waitKey(0)
# save the rotated image to disk
cv2.imwrite('rotated_image.jpg', rotated_image)
```

```
#3program
#Read an image, first apply erosion to the image and then subtract the result
from the original.
#Demonstrate the differencein the edge image if you use dilation instead of
erosion

# Python program to demonstrate erosion and
# dilation of images.
import cv2
import numpy as np

# Reading the input image
img = cv2.imread('input.jpg', 0)

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5, 5), np.uint8)

# The first parameter is the original image,
# kernel is the matrix with which image is
# convolved and third parameter is the number
# of iterations, which will determine how much
# you want to erode/dilate a given image.
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)

cv2.imshow('Input', img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)

cv2.waitKey(0)
```

```
#4 program
#Read an image and extract and display low-level features
#such as edges, textures usingfiltering techniques
#all features in one pic

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

img = cv2.imread("car.jpg", cv2.IMREAD_GRAYSCALE)
lap = cv2.Laplacian(img, cv2.CV_64F, ksize=3)
lap = np.uint8(np.absolute(lap))
sobelX = cv2.Sobel(img, cv2.CV_64F, 1, 0)
sobelY = cv2.Sobel(img, cv2.CV_64F, 0, 1)
edges = cv2.Canny(img,100,200)

sobelX = np.uint8(np.absolute(sobelX))
sobelY = np.uint8(np.absolute(sobelY))

sobelCombined = cv2.bitwise_or(sobelX, sobelY)

titles = ['image', 'Laplacian', 'sobelX', 'sobelY', 'sobelCombined', 'Canny']
images = [img, lap, sobelX, sobelY, sobelCombined, edges]
for i in range(6):
    plt.subplot(2, 3, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

```
#5 program
#Demonstrate enhancing and segmenting low contrast 2D images

import cv2
import matplotlib.pyplot as plt

# Load the image
img = cv2.imread('images 5.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply histogram equalization
equ = cv2.equalizeHist(gray)

# Apply Otsu's thresholding to segment the image
_, thresh = cv2.threshold(equ, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Display the images
plt.subplot(131), plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(132), plt.imshow(equ, cmap='gray')
plt.title('Enhanced Image'), plt.xticks([]), plt.yticks([])
plt.subplot(133), plt.imshow(thresh, cmap='gray')
plt.title('Segmented Image'), plt.xticks([]), plt.yticks([])
plt.show()
```