```
#Program-7 Write a Program to read a digital image. Split and display image into 4 #quadrants, up, down, right and left.
import cv2
from matplotlib import pyplot as plt
# Load image
image = cv2.imread('D:\\puppy.jpg')
# create figure
plt.figure(figsize=(7, 8))
# setting values to rows and column variables
rows = 3; columns = 2
#Get the height and width of the image
(h, w) = image.shape[:2]
#Converting BGR to RGB
image=image[:,:,::-1]
# get center of image and store in (cX,cY)
 (cX, cY) = (w // 2, h // 2)
# crop the image into four parts

topLeft = image[0:cY, 0:cX] # top left
topRight = image[0:cY, cX:w] #top right
bottomLeft = image[cY:h, 0:cX] # bottom left.
bottomRight = image[cY:h, cX:w] # bottom right.

# Adds a subplot at the 1st position and Display original image
plt.subplot(rows, columns, 1)
plt.imshow(image)
plt.axis('off')
plt.title("Original")
# Adds a subplot at the 2nd position and Display top left image
plt.subplot(rows, columns, 3)
plt.imshow(topLeft)
plt.axis('off')
plt.title("topLeft")
# Adds a subplot at the 3rd position and Display top right image
plt.subplot(rows, columns, 4)
plt.imshow(topRight)
plt.axis('off')
plt.title("topRight")
# Adds a subplot at the 4th position and Display bottom left image
plt.subplot(rows, columns, 5)
plt.imshow(bottomLeft)
plt.axis('off')
plt.title("bottomLeft")
# Adds a subplot at the 4th position and Display bottom Right image
plt.subplot(rows, columns, 6)
plt.imshow(bottomRight)
plt.axis('off')
plt.title("bottomRight")
```

```
# Program-8 program to show rotation, scaling, and translation on an image
#Python program to explain cv2.rotate() method, cv2.resize(),translate

import cv2
from matplotlib import pyplot as plt
import numpy as np
plt.figure(figsize=(20, 10))
rows = 2
columns = 2
# Reading an image and convert to rgb
src = cv2.imread('D:\\puppy.jpg')
src=src[:,:,::-1]
# Adds a subplot at the 1st position and display original image
plt.subplot(rows, columns, 1)
plt.imshow(src)
plt.axis('off')
plt.title("Original")

# Using cv2.rotate() method
# Using cv2.ROTATE_90_CLOCKWISE rotate  by 90 degrees clockwise
rot = cv2.rotate(src, cv2.ROTATE_90_CLOCKWISE)
# Adds a subplot at the 2nd position and display rotated image
plt.subplot(rows, columns, 2)
plt.imshow(rot)
plt.axis('off')
plt.title("Rotated")

#to resize create new dimension and use cv2.resize()
(h,w)=src.shape[:2]
newdim=(100,h)
img_shrinked = cv2.resize(src, newdim, interpolation=cv2.INTER_AREA)
# Adds a subplot at the 3rd position and display rotated image
plt.subplot(rows, columns, 3)
plt.imshow(img_shrinked)
plt.axis('off')
plt.title("scaled")

# shift the image (dx=25)25 pixels to the right and (dy=50)50 pixels down
M = np.float32([[1, 0, 25], [0, 1, 50]])
shifted = cv2.warpAffine(src, M, (w,h))
# Adds a subplot at the 4th position and display translated image
plt.subplot(rows, columns, 4)
plt.imshow(shifted)
plt.axis('off')
plt.title("Translated")
```

#9. Read an image and extract and display low-level features such as edges, textures using #filtering techniques.

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

# Load the image

img = cv2.imread("D:\\puppy.jpg")
plt.figure(figsize=(7, 8))
plt.subplot(2, 2, 1)

# Display the original image
plt.imshow(img[:,:,::-1)
plt.axis('off')
plt.title("Original Image")

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

# Edge detection
edges = cv2.Canny(gray, 100, 200)  # Use Canny edge detector

# Texture extraction
kernel = np.ones((5, 5), np.float32) / 25  # Define a 5x5 averaging kernel
texture = cv2.filter2D(gray, -1, kernel)  # Apply the averaging filter for texture extraction

# Display the edges, and texture

plt.subplot(2, 2, 2)
plt.imshow(edges)
plt.axis('off')
plt.title("Edges")

plt.subplot(2, 2,3)
plt.imshow(texture)
plt.axis('off')
plt.title("Texture")
```

#Program-10 Write a program to blur and smoothing an image.
#Smoothing an image using an average blur.
#Notice as how the kernel size increases, the image becomes progressively more blurred.

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

plt.figure(figsize=(7, 8))

# Reading an image and coverting to RGB
src = cv2.imread('D:\\puppy.jpg')
src=src[:,:,::-1]

#Display Original Image
plt.subplot(2, 2, 1)
plt.imshow(src)
plt.axis('off')
plt.title("Original")

i=2
kernelSizes = [(3, 3), (9, 9), (15, 15)]
# loop over the kernel sizes
for (kX, kY) in kernelSizes:
    # apply an "average" blur to the image using the current kernel size and display
    blurred = cv2.blur(src, (kX, kY))
    plt.subplot(2, 2, i)
    plt.imshow(blurred)
    plt.axis('off')
    plt.title("blurred ")
    i+=1
```

```
#Program-11 Write a program to contour an image.
# Grayscale
import cv2
from matplotlib import pyplot as plt
import numpy as np
plt.figure(figsize=(7, 8))

# Reading an image
src = cv2.imread('D:\\puppy.jpg')

gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

# Find Canny edges
edged = cv2.Canny(gray, 30, 200)

contours, hierarchy = cv2.findContours(edged,cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

plt.subplot(1, 2, 1)

plt.imshow(edged)
plt.axis('off')
plt.title("Canny Edges After Contouring")

print("Number of Contours found = " + str(len(contours)))

# Draw all contours
# -1 signifies drawing all contours,(0,255,0) represents color,1 represents thickness
cv2.drawContours(src, contours, -1, (0, 255, 0), 1)


plt.subplot(1, 2, 2)
imgwithcontour=src[:,:,::-1]
# showing image
plt.imshow(imgwithcontour)
plt.axis('off')
plt.title("Contours")
```

#Program-12 Write a program to detect a face/s in an image

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

# Reading an image
src = cv2.imread(D:\\a1.jpg')

gray_image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)

face = face_classifier.detectMultiScale(
    gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(40, 40)
)

for (x, y, w, h) in face:
    cv2.rectangle(src, (x, y), (x + w, y + h), (0, 255, 0), 4)

img_rgb = src[:,:,::-1]

plt.figure(figsize=(7,8))
plt.imshow(img_rgb)
plt.axis('off')
```