7:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
img = cv2.imread("C:\\Users\\CAEDLAB1\\Desktop\\manoj\\image1.jpg")

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

# Split the image into four quadrants
quad1 = img[:height//2, :width//2]
quad2 = img[:height//2, width//2:]
quad3 = img[height//2:, :width//2]
quad4 = img[height//2:, width//2:]
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(quad1)
plt.title("1")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(quad2)
plt.title("2")
plt.axis("off")

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(quad3)
plt.title("3")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(quad4)
plt.title("4")
plt.axis("off")

plt.show()
```

8:
```
import cv2

def translate_image(image, dx, dy):
    rows, cols = image.shape[:2]
    translation_matrix = np.float32([[1, 0, dx], [0, 1, dy]])
    translated_image = cv2.warpAffine(image, translation_matrix, (cols, rows))
    return translated_image

# Read the image
image = cv2.imread("C:\\Users\\CAEDLAB1\\Desktop\\manoj\\image1.jpg")

# Get image dimensions
height, width = image.shape[:2]

# Calculate the center coordinates of the image
center = (width // 2, height // 2)
rotation_value = int(input("Enter the degree of Rotation:"))
scaling_value = int(input("Enter the zooming factor:"))
# Create the 2D rotation matrix
rotated = cv2.getRotationMatrix2D(center=center, angle=rotation_value, scale=1)
rotated_image = cv2.warpAffine(src=image, M=rotated, dsize=(width, height))
scaled = cv2.getRotationMatrix2D(center=center, angle=0, scale=scaling_value)
scaled_image = cv2.warpAffine(src=rotated_image, M=scaled, dsize=(width, height))
h = int(input("How many pixels you want the image to be translated horizontally? "))
v = int(input("How many pixels you want the image to be translated vertically? "))
translated_image = translate_image(scaled_image, dx=h, dy=v)
cv2.imwrite('Final_image.png', translated_image)
```

11:

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

# Load the image from file
image_path = "C:\\Users\\CAEDLAB1\\Desktop\\manoj\\image1.jpg"
image = cv2.imread(image_path)

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

# Apply thresholding
_, binary_image = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# Find contours
contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

```
# Draw all contours on a copy of the original image
image_with_contours = image.copy()
cv2.drawContours(image_with_contours, contours, -1, (0, 255, 0), 3)

# Convert BGR image to RGB for displaying with matplotlib
image_rgb = cv2.cvtColor(image_with_contours, cv2.COLOR_BGR2RGB)

# Display the original image and the image with contours side by side using matplotlib
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(image_rgb)
plt.title('Image with Contours')
plt.axis('off')

plt.tight_layout()
plt.show()
```

10:

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

img =
cv2.imread("C:\\Users\\CAEDLAB1\\Desktop\\manoj\\image1.jpg",cv2.IMREAD_GRAYSCALE)
image_array = np.array(img)
print(image_array)
def sharpen():
  return np.array([
[1,1,1],[1,1,1],[1,1,1]
  ])
def filtering(image, kernel):
    m, n = kernel.shape
    if (m == n):
        y, x = image.shape
        y = y - m + 1 # shape of image - shape of kernel + 1
        x = x - m + 1
        new_image = np.zeros((y,x))
        for i in range(y):
            for j in range(x):
                new_image[i][j] = np.sum(image[i:i+m, j:j+m]*kernel)
    return new_image
# Display the original and sharpened images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image_array,cmap='gray')
plt.title("Original Grayscale Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(filtering(image_array, sharpen()),cmap='gray')
plt.title("Blurred Image")
plt.axis("off")
plt.show()
```

9:

```
import cv2
import numpy as np
# Load the image
image_path = "C://Users/CSELAB2/Desktop/pictures/6.jpeg" # Replace with the path to your
image
img = cv2.imread(image_path)
# 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 original image, edges, and texture
cv2.imshow('Original Image', img)
cv2.imshow('Edges', edges)
cv2.imshow('Texture', texture)
# Wait for a key press and then close all windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```