# Barrackpore Rastraguru Surendranath College

## B.Sc. (Hons.) in Computer Science
## Semester-VI

**B.Sc. Computer Science Honours Semester-VI**
**Examination 2023**

| | |
|---|---|
| Name : | LABANI DAS |
| College Roll No : | 2001055 |
| Semester : | SEM-VI |
| WBSU Reg. No. : | 1032021100062 |
| Subject : | Computer Science Hons. (CMSA) |
| Paper Code : | CMSADSE05P |
| Paper Name : | Digital Image Processing Assignments |

# INDEX

**WRITE A PYTHON PROGRAM TO :**

## PROGRAM NUMBER: 1

Read and display an image.

**Program code:**

```
# importing cv2
import cv2

# Using cv2.imread() method
# Using 0 to read image in grayscale mode
img = cv2.imread("images/Catching.JPG", 0)
# Displaying the image
cv2.imshow('image', img)

# Using 1 to read image in color mode
img = cv2.imread("images/Catching.jpg", 1)
# Displaying the image
cv2.imshow('image1', img)

# Wait for the user to press a key
cv2.waitKey(0)

# Close all windows
cv2.destroyAllWindows()
```

**Program output:**

*color mode*

*grayscale mode*

## PROGRAM NUMBER: 2

Write down the dimensions of an image

**Program code:**

```
#Display Dimensions of image

# importing cv2
import cv2

img = cv2.imread("images/Catching.jpg")

# get dimensions of image
dimensions = img.shape

# height, width, number of channels in image
height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]

print('Image Dimension    : ',dimensions)
print('Image Height       : ',height)
print('Image Width        : ',width)
print('Number of Channels : ',channels)
print('Number of Pixel : ',img.size)
```

**Program output:**

*Image*

```
Image Dimension    :  (1944, 2592, 3)
Image Height       :  1944
Image Width        :  2592
Number of Channels :  3
Number of Pixel :  15116544
```

*dimensions*

## PROGRAM NUMBER:3

Display the results of height and width reduction of an image to 30% of its current height and width

### Program code:

```
# importing cv2
import cv2

img = cv2.imread("images/Catching.jpg")

print('Original Dimensions : ',img.shape)

scale_percent = 30 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

print('Resized Dimensions : ',resized.shape)

cv2.imshow("Resized image", resized)


# Wait for the user to press a key
cv2.waitKey(0)

# Close all windows
cv2.destroyAllWindows()
```

### Program output:

*Original size*

*reduced size*

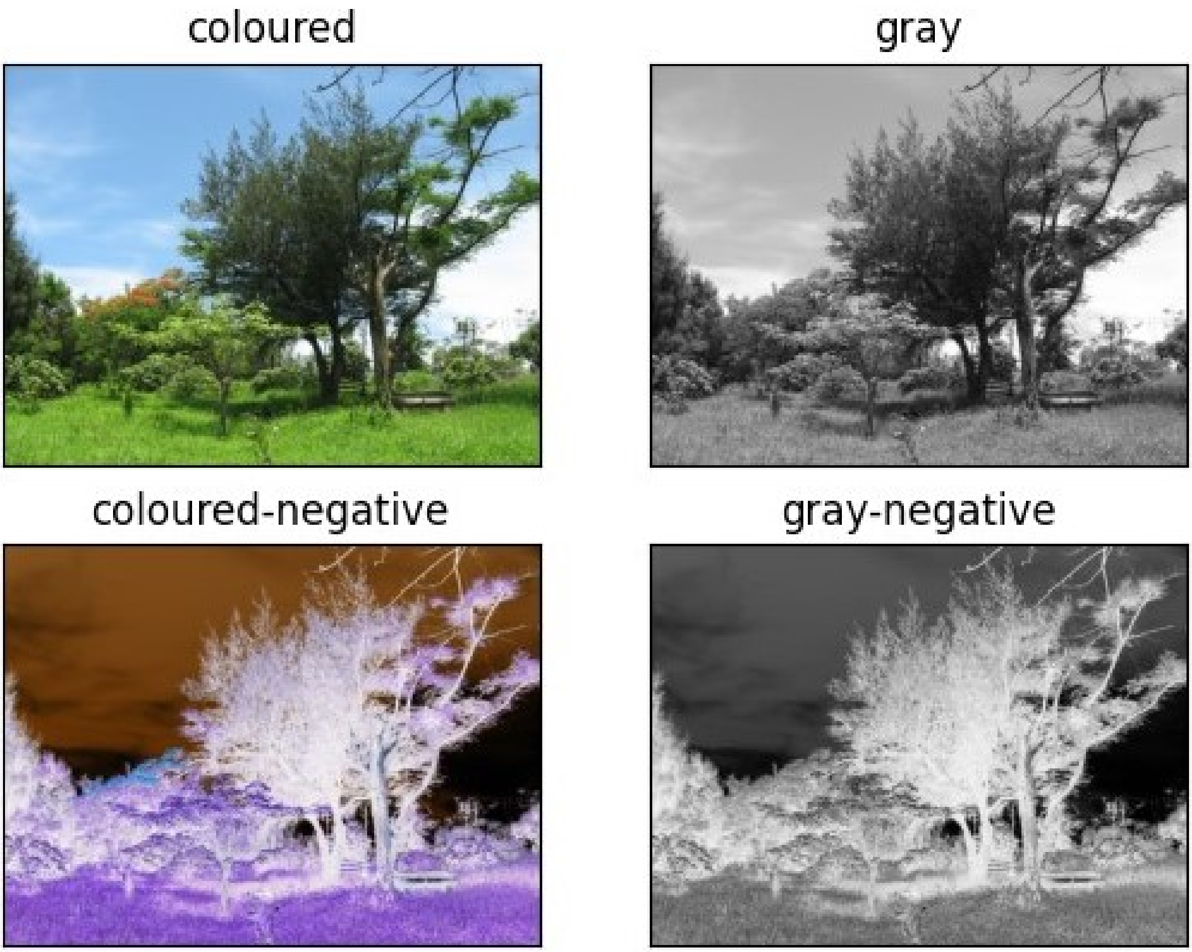
## PROGRAM NUMBER:4

Display the color negative and grayscale negative of a color image.

### Program code:

```
# importing cv2
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("images/Catching.jpg")
gray= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
colored_negative = abs(255-img)
gray_negative = abs(255-gray)

imgs = [img, gray, colored_negative, gray_negative]
title = ['coloured', 'gray', 'coloured-negative', 'gray-negative']


plt.subplot(2, 2, 1)
plt.title(title[0])
plt.imshow(imgs[0])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 2, 2)
plt.title(title[1])
plt.imshow(imgs[1], cmap='gray')
plt.xticks([])
plt.yticks([])

plt.subplot(2, 2, 3)
plt.title(title[2])
plt.imshow(imgs[2])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 2, 4)
plt.title(title[3])
plt.imshow(imgs[3], cmap='gray')
plt.xticks([])
plt.yticks([])

plt.show()
```

**Program output:**



## PROGRAM NUMBER:5

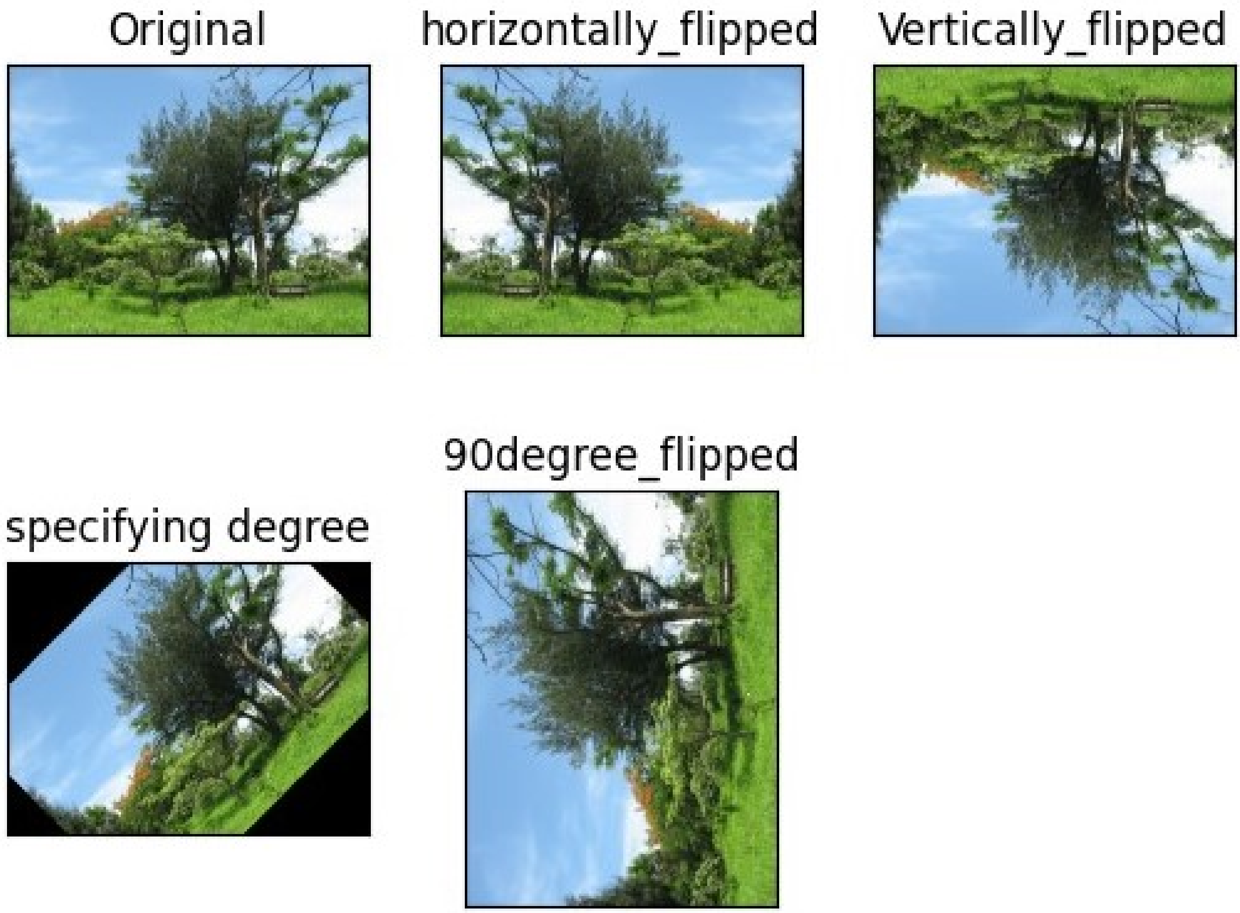
Display the results of flipping an image vertically and horizontally. Also rotate the image by 45 degrees and 90 degrees.

### Program code:

```
# importing cv2
import cv2
import matplotlib.pyplot as plt
from PIL import Image


img = Image.open("images/Catching.jpg")
hori_flippedImage = img.transpose(Image.FLIP_LEFT_RIGHT)
Vert_flippedImage = img.transpose(Image.FLIP_TOP_BOTTOM)
degree_flippedImage = img.rotate(45)
degree90_flippedImage = img.transpose(Image.ROTATE_90)


imgs = [img, hori_flippedImage, Vert_flippedImage, degree_flippedImage,
degree90_flippedImage]
title = ['Original', 'horizontally_flipped', 'Vertically_flipped', 'specifying
degree', '90degree_flipped']


plt.subplot(2, 3, 1)
plt.title(title[0])
plt.imshow(imgs[0])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 3, 2)
plt.title(title[1])
plt.imshow(imgs[1])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 3, 3)
plt.title(title[2])
plt.imshow(imgs[2])
plt.xticks([])
plt.yticks([])

plt.subplot(2, 3, 4)
plt.title(title[3])
plt.imshow(imgs[3])
plt.xticks([])
plt.yticks([])
plt.subplot(2, 3, 5)
```

```
plt.title(title[4])
plt.imshow(imgs[4])
plt.xticks([])
plt.yticks([])


plt.show()
```

**Program output:**



## PROGRAM NUMBER:6

Display the results of down-sampling using 'face' and 'rose' image

## Program code:

```
from PIL import Image

def downsample_image(image_path, output_path, factor):
    # Open the image
    image = Image.open(image_path)

    # Calculate the new width and height based on the downsampling factor
    width = image.width // factor
    height = image.height // factor

    # Downsample the image using the 'ANTIALIAS' filter for better quality
    downscaled_image = image.resize((width, height), Image.ANTIALIAS)

    downscaled_image.show()

def show_downsampled_image(file_name):
    input_image_path = f"Images/{file_name}"
    output_image_path = f"6/{file_name.split('.')[0]}_downsampled.jpg"
    downsampling_factor = 2

    #  downsample and save image
    downsample_image(input_image_path, output_image_path, downsampling_factor)

# face image
show_downsampled_image("face.jpg")
# rose image
show_downsampled_image("rose-512.bmp")
```

## Program output:

Original image

down-sampled image

Original image

down-sampled image

## PROGRAM NUMBER:7

Display the effects of quantization using ‘face’ and ‘rose’ image.

### Program code:

```
from PIL import Image

def quantize_image(image_path, output_path, num_colors):
    # Open the image
    image = Image.open(image_path)

    # Apply quantization by reducing the number of colors
    quantized_image = image.quantize(num_colors)

    #show image
    quantized_image.show()


def showQuantizedimage(file_name):
    input_image_path = f"images/{file_name}"
    output_image_path = f"7/{file_name.split('.')[0]}_quantized.jpg"
    num_colors = 20
```

```
    quantize_image(input_image_path, output_image_path, num_colors)

# face image
showQuantizedimage("face.jpg")
# rose image
showQuantizedimage("rose-512.bmp")
```

**Program output:**

*Original image*

*quantized image*

*Original image*

*quantized image*

## PROGRAM NUMBER:8

Calculate mean value of a grayscale image. Convert the grayscale image to a binary image using the mean value as the threshold. Use ‘house’ image to show the results.

### Program code:

```
import numpy as np
from PIL import Image

# Open the image and convert it to grayscale
image = Image.open('images/house.jpg')

# Convert the image to a NumPy array
image_array = np.array(image)

# Calculate the mean value
mean_value = np.mean(image_array)

# show mean value
print("Mean value:", mean_value)

# Create a binary image by applying the threshold
# if image_array >= mean_value TRUE value assigned to that pixel is 255
# if image_array >= mean_value FALSE value assigned to that pixel is 0
binary_image = np.where(image_array >= mean_value, 255, 0).astype(np.uint8)

# Convert the NumPy array back to an image
binary_image = Image.fromarray(binary_image)

# show the binary image
binary_image.show()
```

### Program output:

*Original image*

*binary image*

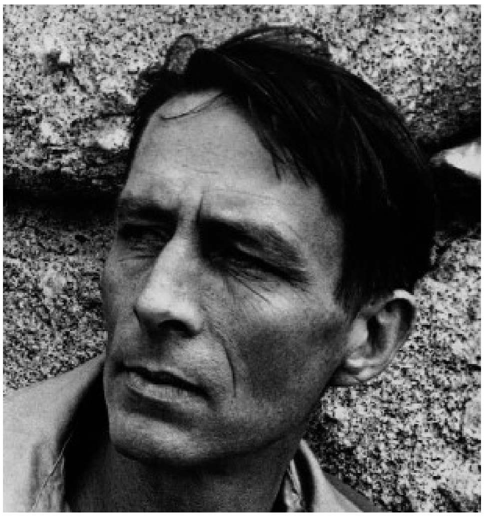
## PROGRAM NUMBER:9

Display the smoothing effects of 3×3 and 5×5 mean filters using 'man' image.

**Program code:**

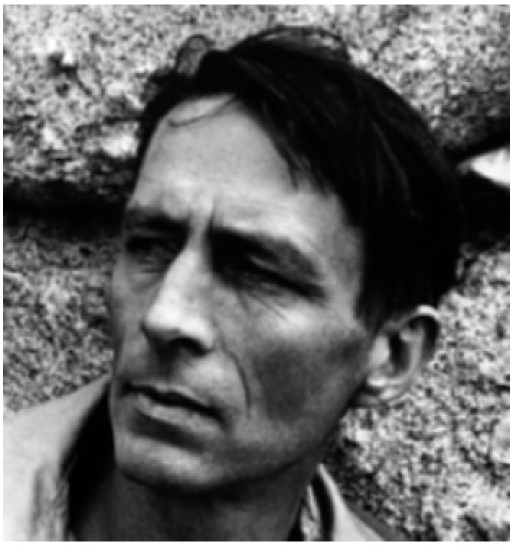
```
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

def apply_mean_filter(image_path, kernel_size, title):
    # Open the image using Pillow
    image = Image.open(image_path)

    # Convert the image to a NumPy array
    image_array = np.array(image)

    # Apply the mean filter
    filtered_image_array = cv2.blur(image_array, (kernel_size, kernel_size))

    # convert image array to image
    filtered_image = Image.fromarray(filtered_image_array)

    # Display the image with the specified title
    plt.imshow(filtered_image, cmap='gray')
    plt.title(title)
    plt.axis('off')
    plt.show()


# man image path
input_image_path = 'images/man.gif'
kernel_size_3x3 = 3
kernel_size_5x5 = 5

apply_mean_filter(input_image_path, kernel_size_3x3, "smoothning using 3x3
mean filter")
apply_mean_filter(input_image_path, kernel_size_5x5,"smoothning using 5x5 mean
filter ")
```

**Program output:**



*original image*

smoothning using 3x3 mean filter



*blurred imahs using 3x3 smoothning mean filter*

smoothning using 5x5 mean filter

*blurred imahs using 5x5 smoothning mean filter*

## PROGRAM NUMBER:10

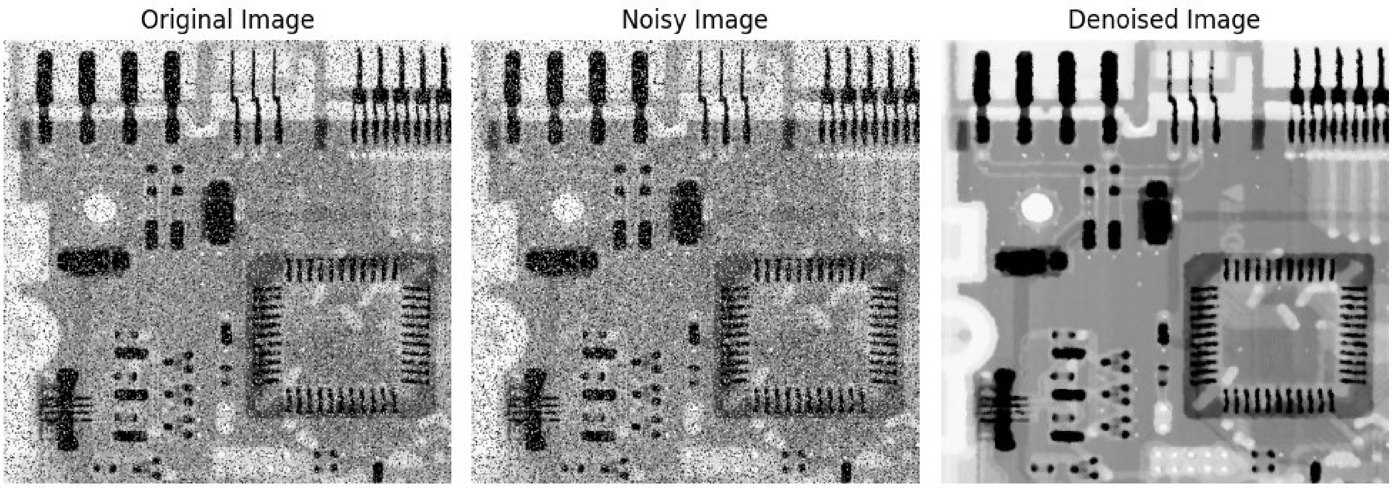
Display the results of salt-and-pepper noise reduction by using median filter on 'Circuit_board_noisy' image.

**Program code:**

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

def reduce_salt_and_pepper_noise(image_path,  kernel_size):
    # Load the image
    image = cv2.imread(image_path)

    # Add salt-and-pepper noise to the image
    noisy_image = add_salt_and_pepper_noise(image)

    # Apply median filter for noise reduction
    # The Median blur operation is similar to the other averaging methods.
    # Here, the central element of the image is replaced by the median of all
the pixels in the kernel area.
    denoised_image = cv2.medianBlur(noisy_image, kernel_size)

    # Display the original, noisy, and denoised images
    titles = ['Original Image', 'Noisy Image', 'Denoised Image']
    images = [image, noisy_image, denoised_image]

    plt.figure(figsize=(10, 5))
    for i in range(3):
        plt.subplot(1, 3, i + 1)
        plt.imshow(cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB))
        plt.title(titles[i])
        plt.axis('off')

    plt.tight_layout()
    plt.show()

def add_salt_and_pepper_noise(image, noise_prob=0.05):
    # Generate a mask for salt-and-pepper noise
    mask = np.random.choice([0, 1, 2], size=image.shape[:2], p=[1 -
noise_prob, noise_prob / 2, noise_prob / 2])

    # Add salt-and-pepper noise to the image
    noisy_image = np.copy(image)
    noisy_image[mask == 1] = 255  # Salt noise
    noisy_image[mask == 2] = 0  # Pepper noise

    return noisy_image
```

```python
# image path
input_image_path = 'images/Circuit_board_noisy.tif'
kernel_size = 5

reduce_salt_and_pepper_noise(input_image_path,  kernel_size)
```

**<u>Program output:</u>**



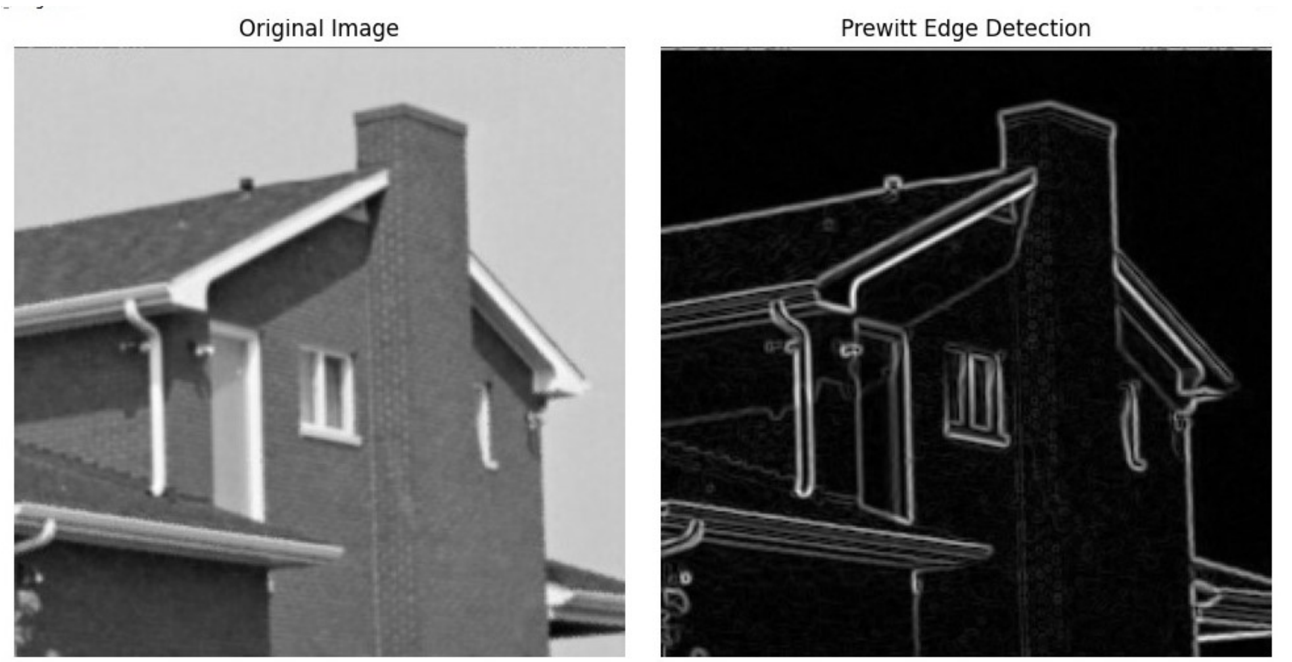
## PROGRAM NUMBER:11

Display the results of edge detection by Prewitt operator, using 'house' image.

**Program code:**

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


def apply_prewitt_operator(image_path):
    # Load the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Apply the Prewitt operator
    gradient_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    gradient_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)

    # Combine the gradients to obtain the edge magnitude
    gradient_magnitude = np.sqrt(np.square(gradient_x) +
np.square(gradient_y))

    # Normalize the gradient magnitude to the range [0, 255]
    gradient_magnitude_normalized = cv2.normalize(gradient_magnitude, None, 0,
255, cv2.NORM_MINMAX, dtype=cv2.CV_8U)

    # Display the original and edge-detected images
    titles = ['Original Image', 'Prewitt Edge Detection']
    images = [image, gradient_magnitude_normalized]

    plt.figure(figsize=(10, 5))
    for i in range(2):
        plt.subplot(1, 2, i + 1)
        plt.imshow(images[i], cmap='gray')
        plt.title(titles[i])
        plt.axis('off')

    plt.tight_layout()
    plt.show()


# image path
input_image_path = 'images/house.jpg'

apply_prewitt_operator(input_image_path)
```

**Program output:**



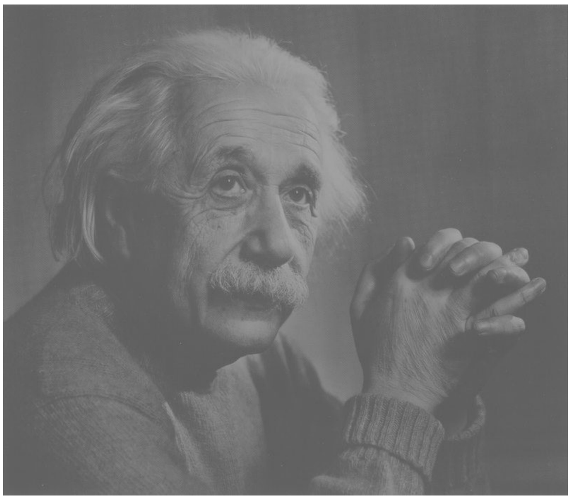
## PROGRAM NUMBER:12

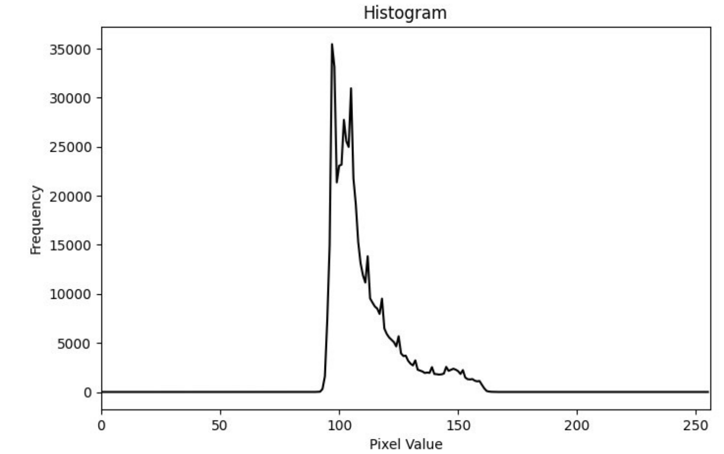
Display the histograms of low-contrast, medium-constrast, and high-constrast images

**Program code:**

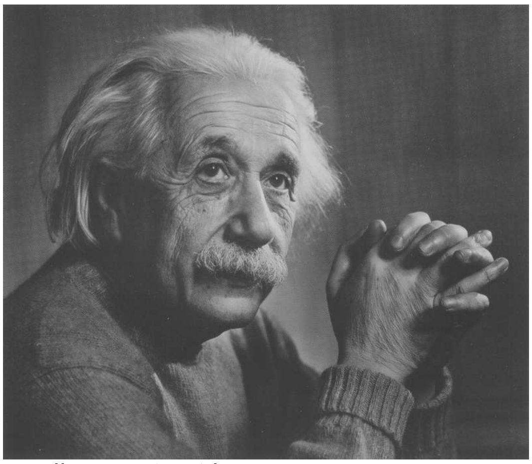
```
import cv2
import matplotlib.pyplot as plt

def display_histogram(image_path):
    # Load the image
    image = cv2.imread(image_path, 0)

    # Calculate the histogram
    histogram = cv2.calcHist([image], [0], None, [256], [0, 256])

    # Display the histogram
    plt.figure(figsize=(8, 5))
    plt.plot(histogram, color='black')
    plt.title('Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')
    plt.xlim([0, 256])
    plt.show()

# Example usage
low_contrast_image_path = 'images/Einstein low contrast.tif'
medium_contrast_image_path = 'images/Einstein med contrast.tif'
high_contrast_image_path = 'images/Einstein high contrast.tif'

display_histogram(low_contrast_image_path)
display_histogram(medium_contrast_image_path)
display_histogram(high_contrast_image_path)
```

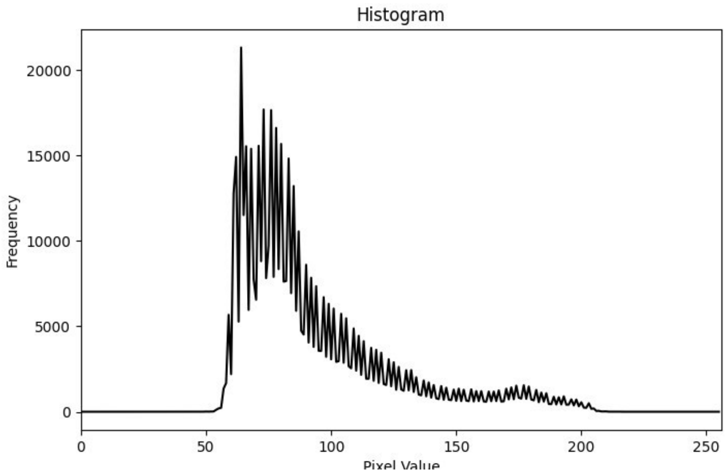
## **Program output:**



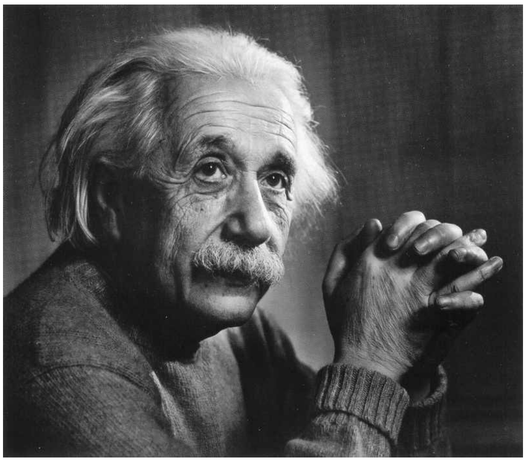


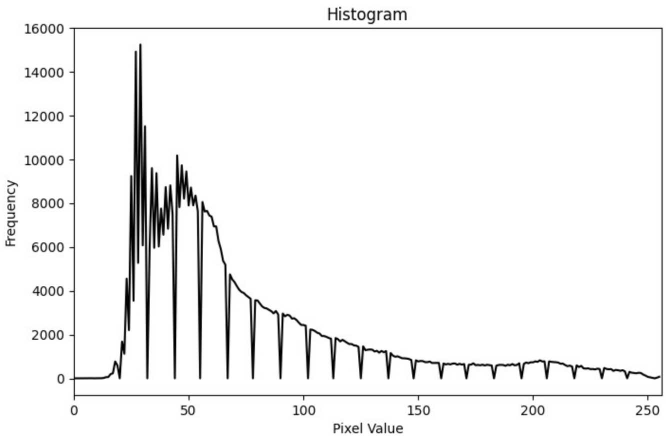

*Low contrast image* *histogram*






*medium contrast image* *histogam*






*high contrast image* *histogam*