EXP NO: 3	IMPLEMENT HISTOGRAM PROCESSING AND EQUALIZATION

AIM:

To understand and implement histogram processing and histogram equalization techniques for image contrast enhancement using OpenCV and Python.

THEORY:

Histogram processing involves analyzing and modifying the distribution of pixel intensities in a digital image. Histogram equalization enhances image contrast by spreading out the most frequent intensity values.

- Histogram: A graph showing the distribution of pixel intensities (0–255).
- Equalization: Enhances contrast by redistributing intensities to span the full range.
- CLAHE (Contrast Limited Adaptive Histogram Equalization): Improves contrast locally in small image regions.

REQUIREMENTS:

- Python 3.x
- OpenCV
- Matplotlib
- NumPy

ALGORITHM:

Step 1: Start

Step 2: Read the input image using cv2.imread().

Step 3: Convert the image to grayscale using cv2.cvtColor() with cv2.COLOR_BGR2GRAY.

Step 4: Compute the histogram using matplotlib's plt.hist() or OpenCV's cv2.calcHist().

Step 5: Apply histogram equalization using cv2.equalizeHist().

Step 6: Apply CLAHE using cv2.createCLAHE() and apply () method (optional).

Step 7: Display original, equalized, and CLAHE images using cv2.imshow().

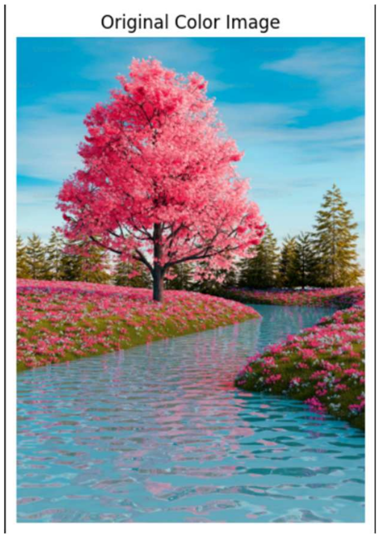
Step 8: Wait for key press using cv2.waitKey(0) and close all windows using cv2.destroyAllWindows().

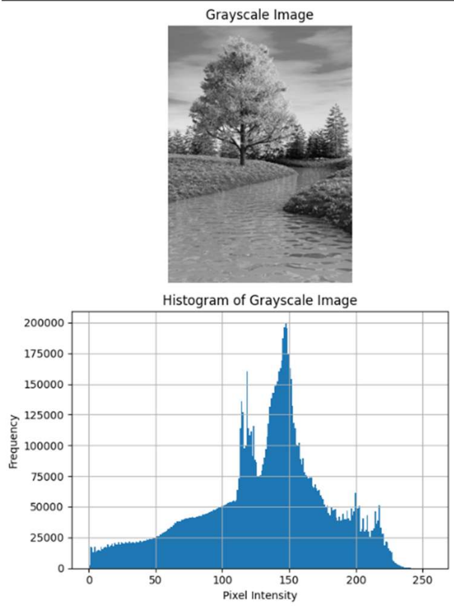
Step 9: End

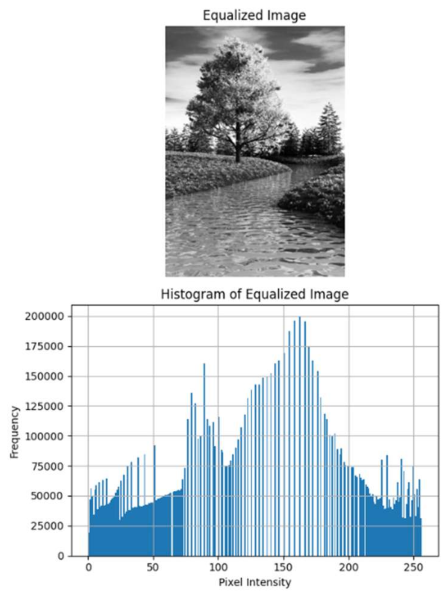
CODE:

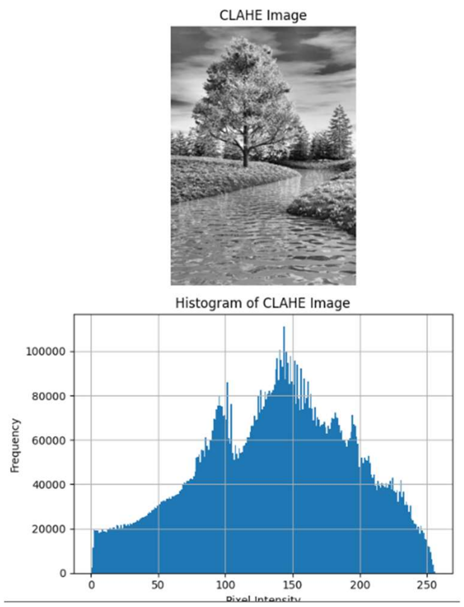
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files

uploaded = files.upload()

filename = list(uploaded.keys())[0]
image = cv2.imread(filename)

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

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

equalized = cv2.equalizeHist(gray)

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
clahe_img = clahe.apply(gray)

plt.figure(figsize=(6, 6))
plt.imshow(image_rgb)
plt.title("Original Color Image")
plt.axis('off')
plt.show()

def show_image_with_histogram(image, title, cmap='gray'):
    plt.figure(figsize=(6, 8))

    plt.subplot(2, 1, 1)
    plt.imshow(image, cmap=cmap)
    plt.title(title)
    plt.axis('off')

    plt.subplot(2, 1, 2)
    plt.hist(image.ravel(), 256, [0, 256])
    plt.title(f"Histogram of {title}")
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
    plt.grid(True)

    plt.tight_layout()
    plt.show()

show_image_with_histogram(gray, "Grayscale Image")
show_image_with_histogram(equalized, "Equalized Image")
show_image_with_histogram(clahe_img, "CLAHE Image")
```

OUTPUT:









RESULT:

The histogram of the image was successfully generated, showing the distribution of pixel intensities. After histogram equalization, the image displayed improved contrast, with dark and bright regions more balanced and visually enhanced.