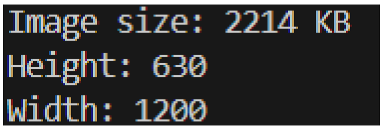
1. Write a program to read an image and display its property.

Source Code:

```
import cv2 as cv
img = cv.imread("P:\\PCA2_10071023015\\nature.jpg")
shape_image = img.shape
if (len(shape_image) == 3):
    height = shape_image[0]
    width = shape_image[1]
    chann = shape_image[2]
print(f"Image size: {(height * width * chann) // 1024} KB")
print(f"Height: {height}")
print(f"Width: {width}")
```

Output:



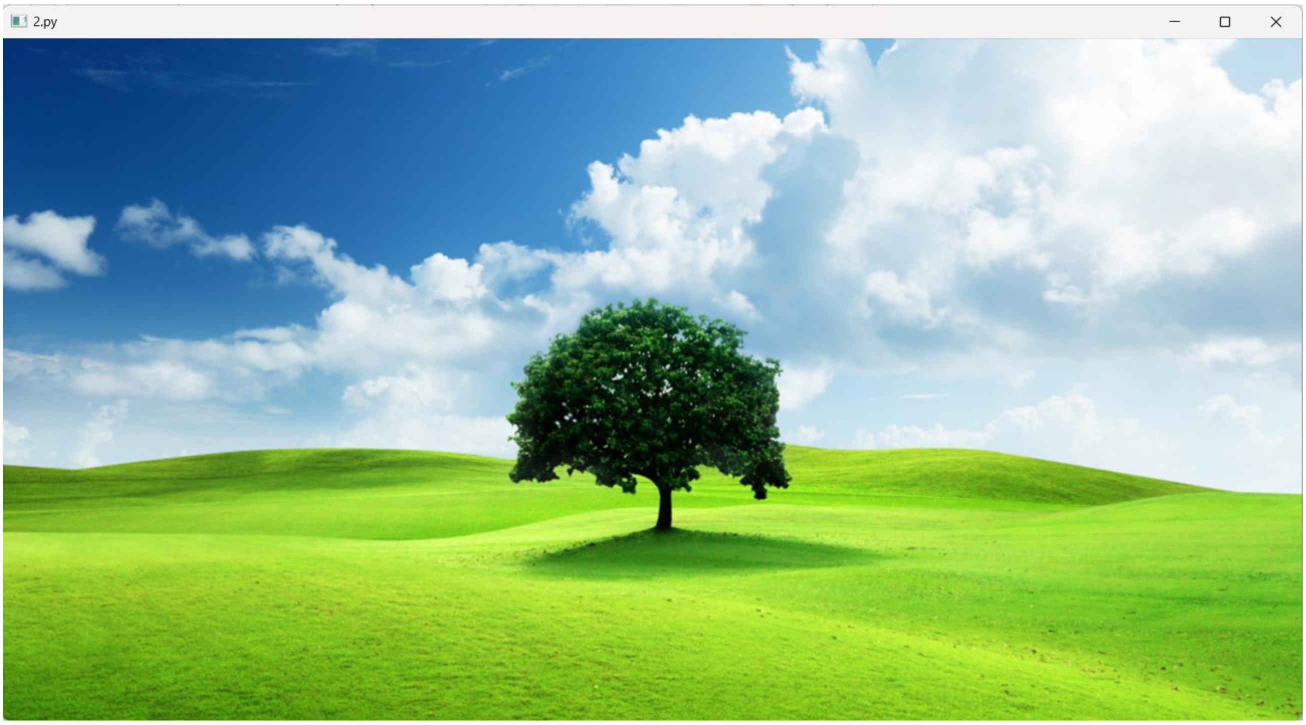
```
Image size: 2214 KB
Height: 630
Width: 1200
```

2. Write a program to display an image:

Source code

```
import cv2 as cv
img = cv.imread("P:\\PCA2_10071023015\\nature.jpg")
if img is not None:
   cv.imshow("2.py", img)
   cv.waitKey(0)
   cv.destroyAllWindows()
else:
   print("Error loading the image. Please check the file path.")
```

output:

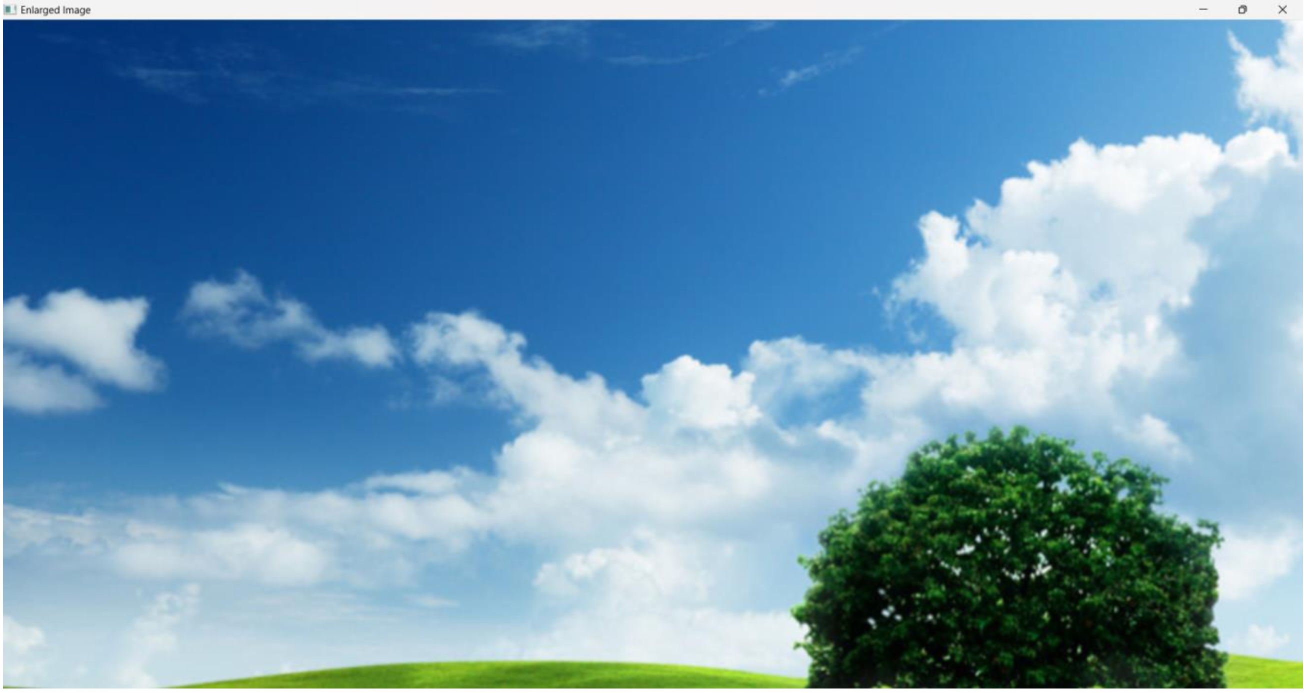


4. Write a program to enlarge an image to its double size

Source Code:

```
import cv2

import numpy as np


image = cv2.imread("P:\\PCA2_10071023015\\nature.jpg")

if image is None:

   print("No file exists")

   exit(1)

original_height, original_width = image.shape[:2]

new_width = original_width * 2

new_height = original_height * 2

enlarged_image = cv2.resize(image, (new_width, new_height), interpolation=cv2.INTER_LINEAR)

cv2.imshow('Original Image', image)

cv2.imshow('Enlarged Image', enlarged_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite('enlarged_image.jpg', enlarged_image)
```

output:

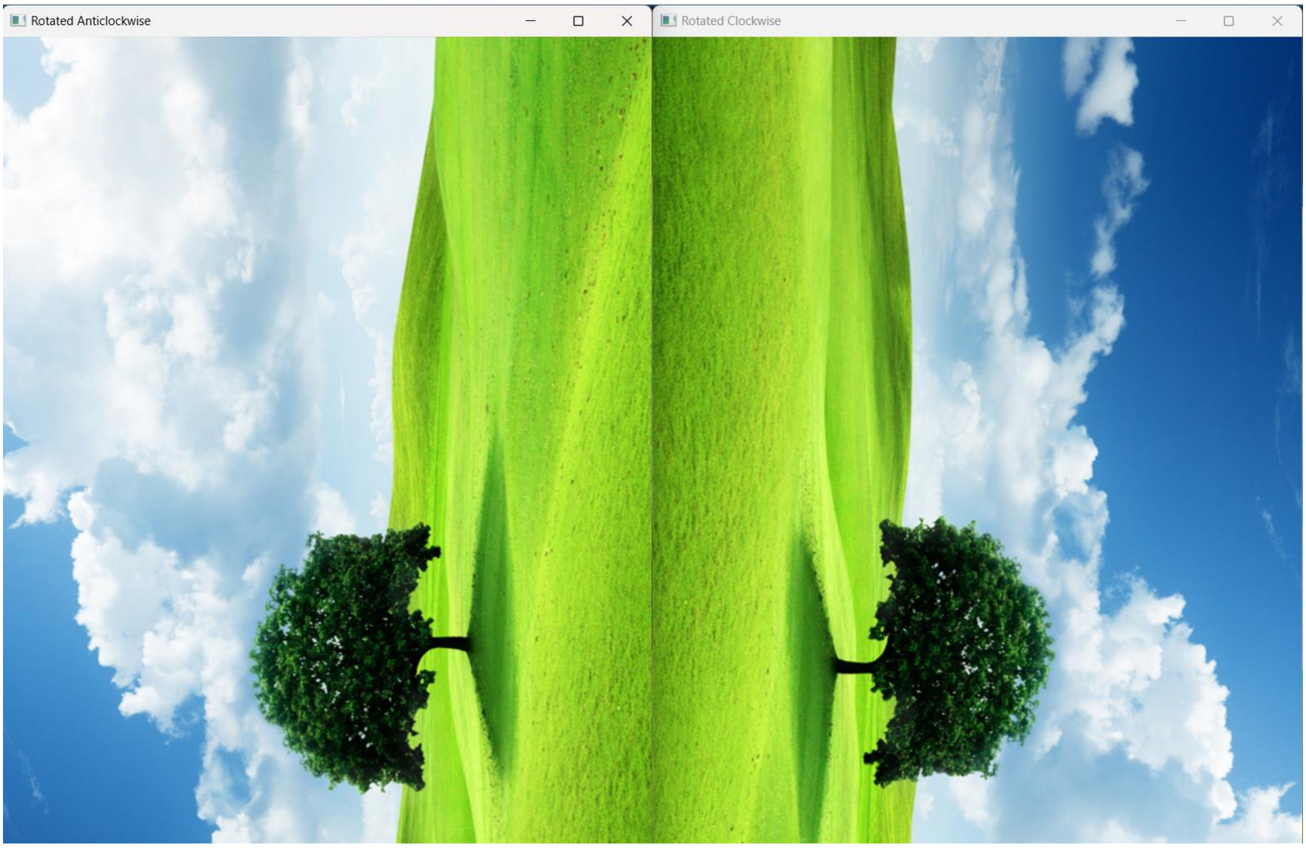


5. Write a program to rotate an image in clockwise and anticlockwise direction.

Souce Code:

```
import cv2
image = cv2.imread("P:\\PCA2_10071023015\\nature.jpg")

if image is None:
   print("Error loading image.")
else:
               rotated_clockwise = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)
              rotated_anticlockwise = cv2.rotate(image, cv2.ROTATE_90_COUNTERCLOCKWISE)

   cv2.imshow('Original Image', image)
   cv2.imshow('Rotated Clockwise', rotated_clockwise)
   cv2.imshow('Rotated Anticlockwise', rotated_anticlockwise)

   cv2.waitKey(0)
   cv2.destroyAllWindows()
```

output:

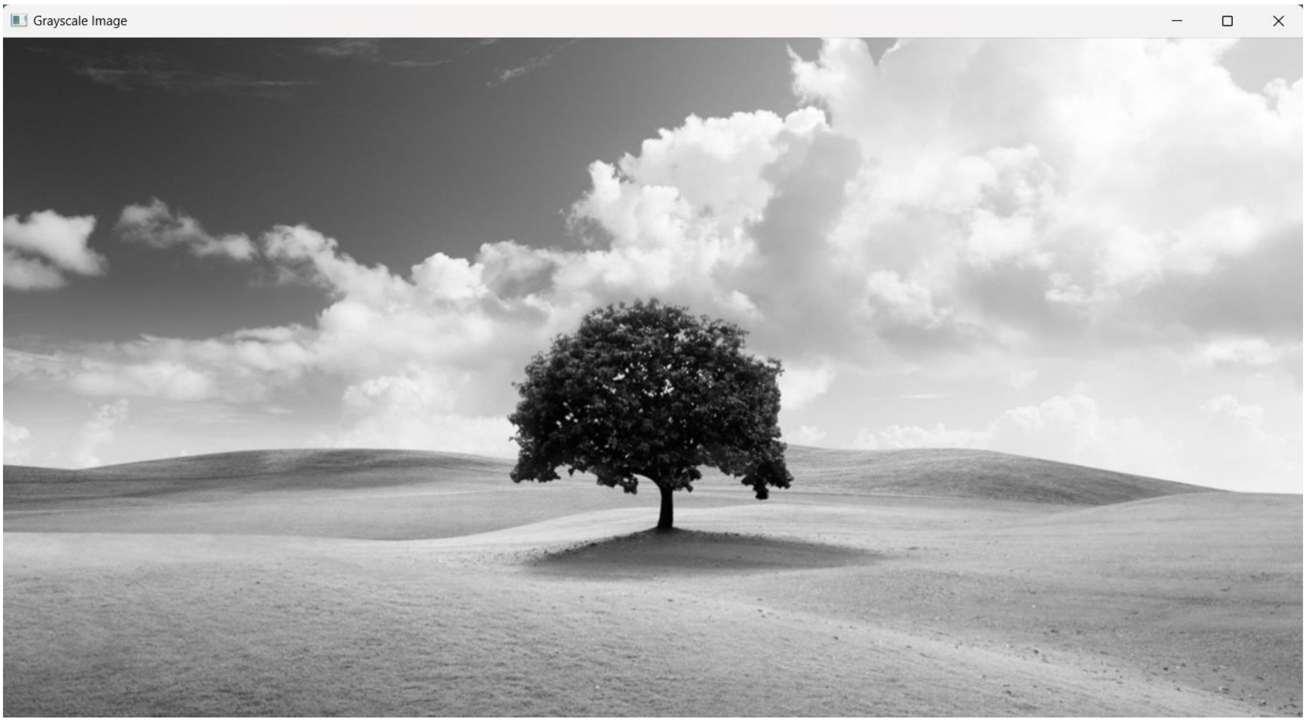


6. Write a program to convert and rgb image to gray scale image.

Source code:

```
import cv2


image = cv2.imread("P:\\PCA2_10071023015\\nature.jpg")


if image is None:
   print("Error loading image.")
else:
   gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   cv2.imshow('Original Image', image)
   cv2.imshow('Grayscale Image', gray_image)
   cv2.imwrite('grayscale_image.jpg', gray_image)
   cv2.waitKey(0)
   cv2.destroyAllWindows()
```

output:

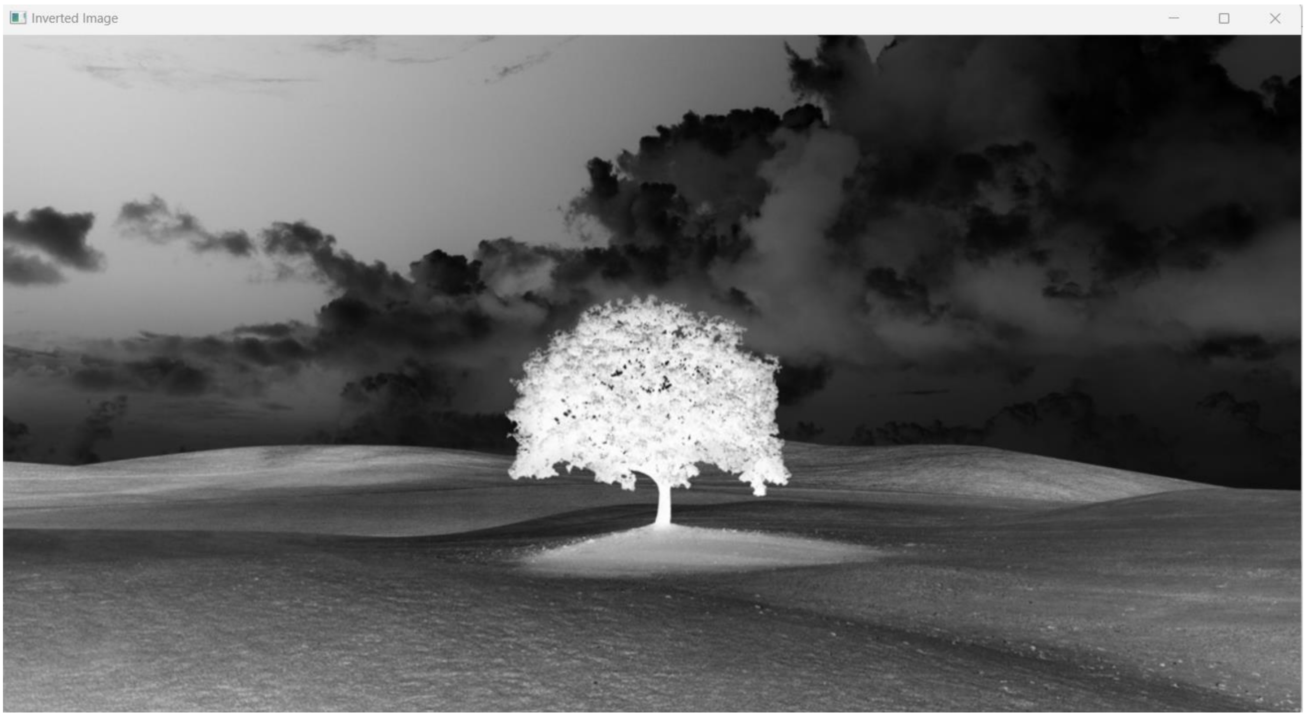


7. Write a program to implement the Basic Gray Level
    - Image Negative

Source Code;

```
import cv2


def invert_image(image):
   return 255 - image


image = cv2.imread("P:\\PCA2_10071023015\\nature.jpg", cv2.IMREAD_GRAYSCALE)


if image is None:
   print("Error loading image.")
else:
   inverted_image = invert_image(image)
   cv2.imshow('Original Image', image)
   cv2.imshow('Inverted Image', inverted_image)
    cv2.waitKey(0)
   cv2.destroyAllWindows()
```

output:

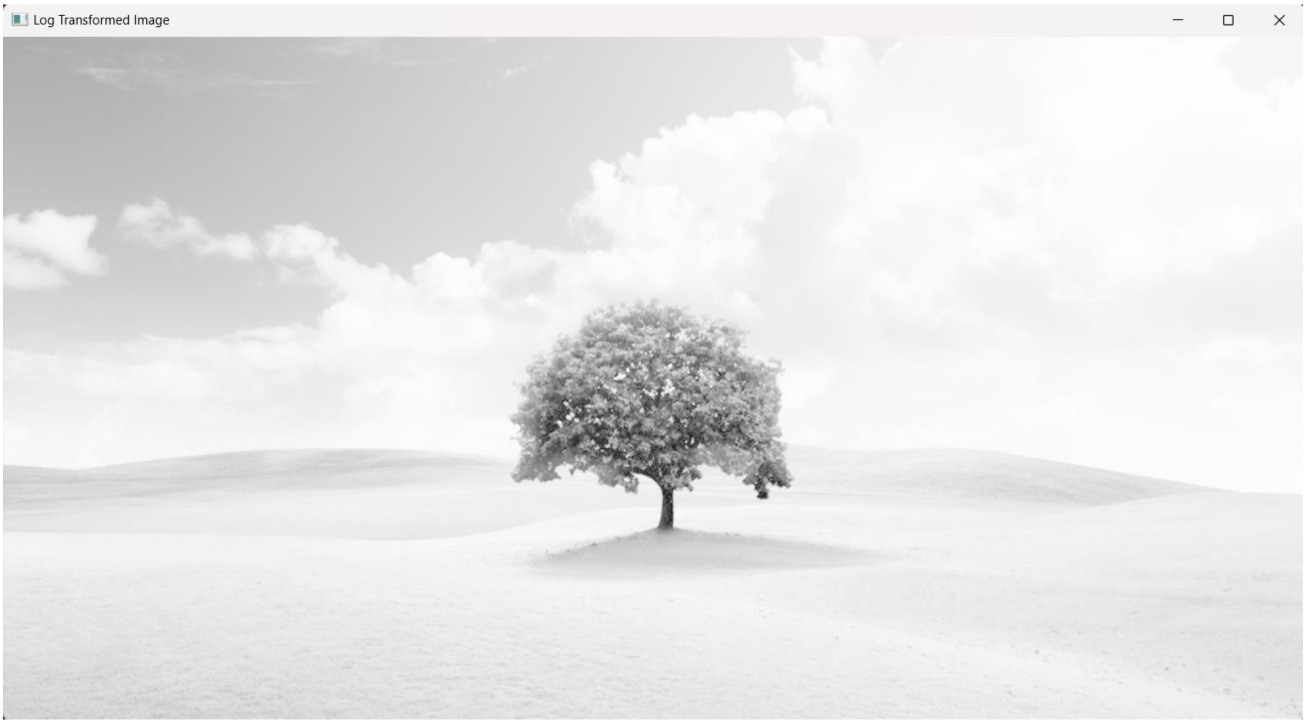


• Log Transformation

```
import cv2

import numpy as np

def log_transform(image):

     image = np.where(image == 0, 1, image)

    image = image.astype(np.float32)

    c = 255 / np.log(1 + np.max(image))

    log_image = c * (np.log(image + 1))

    log_image = np.array(log_image, dtype=np.uint8)

    return log_image


image_path = "P:\\PCA2_10071023015\\nature.jpg"

image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


if image is None:

    print("Error loading image.")

else:
```

```
log_image = log_transform(image)
cv2.imshow('Original Image', image)
cv2.imshow('Log Transformed Image', log_image)

cv2.imwrite('log_transformed_image.jpg', log_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

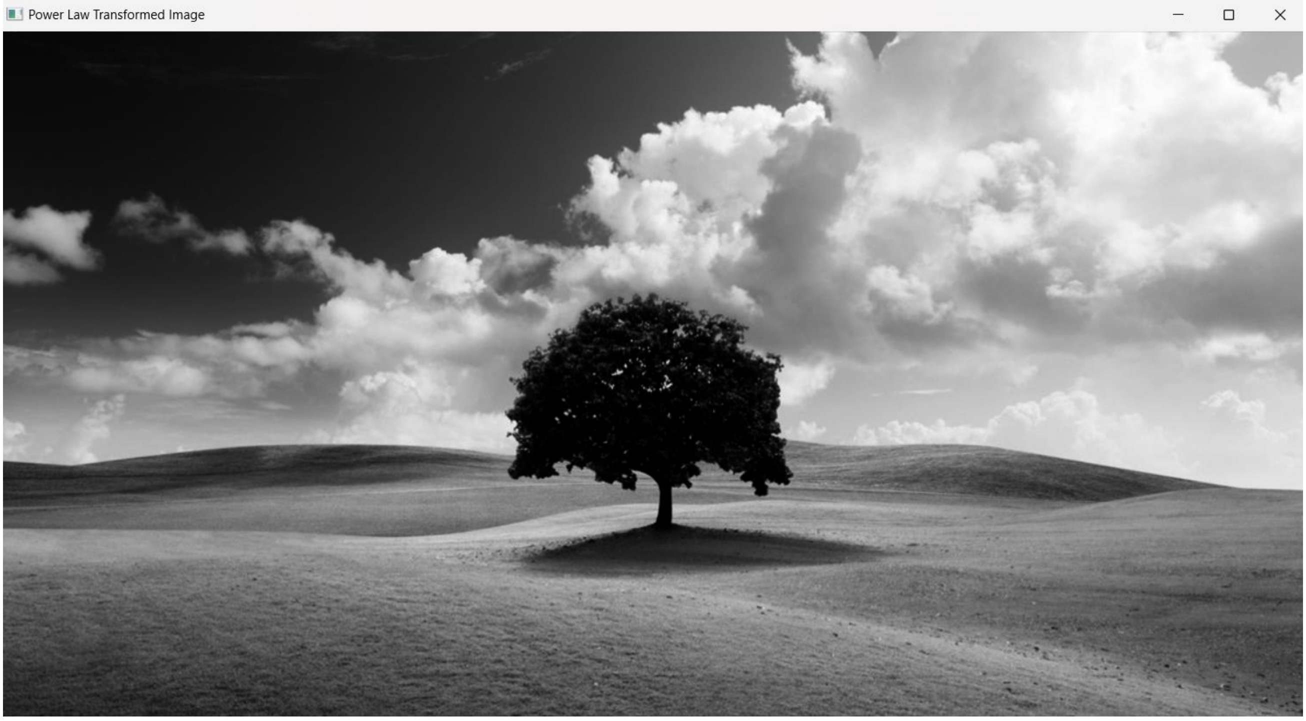
output:



- Power Law Transformation

```
import cv2
import numpy as np

def power_law_transform(image, gamma):

    normalized_img = image / 255.0
    power_law_img = np.power(normalized_img, gamma)
    power_law_img = np.uint8(power_law_img * 255)
    return power_law_img
image_path = "P:\\PCA2_10071023015\\nature.jpg"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

if image is None:
    print("Error loading image.")
else:
    gamma = 2.0
    power_law_image = power_law_transform(image, gamma)
    cv2.imshow('Original Image', image)
    cv2.imshow('Power Law Transformed Image', power_law_image)

    cv2.imwrite('power_law_transformed_image.jpg', power_law_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

```

output:



- Piecewise Linear Transformation (Contrast Stretching)

```
import cv2
import numpy as np

def piecewise_linear_transform(image, low, high):
    normalized_img = image / 255.0
    low = max(0, low)
    high = min(255, high)

    def piecewise_linear(x):
        return np.piecewise(x, [x < low, (low <= x) & (x <= high), x > high], [0, lambda x: ((x - low) / (high - low)) * 255, 255])

    piecewise_img = piecewise_linear(normalized_img)
    piecewise_img = np.uint8(piecewise_img)

    return piecewise_img

image_path = "P:\\PCA2_10071023015\\nature.jpg"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

if image is None:
    print("Error loading image.")
else:

    low = 50
    high = 200

    piecewise_image = piecewise_linear_transform(image, low, high)

    cv2.imshow('Original Image', image)
    cv2.imshow('Piecewise Linear Transformed Image', piecewise_image)

    cv2.imwrite('piecewise_linear_transformed_image.jpg',
piecewise_image)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

```

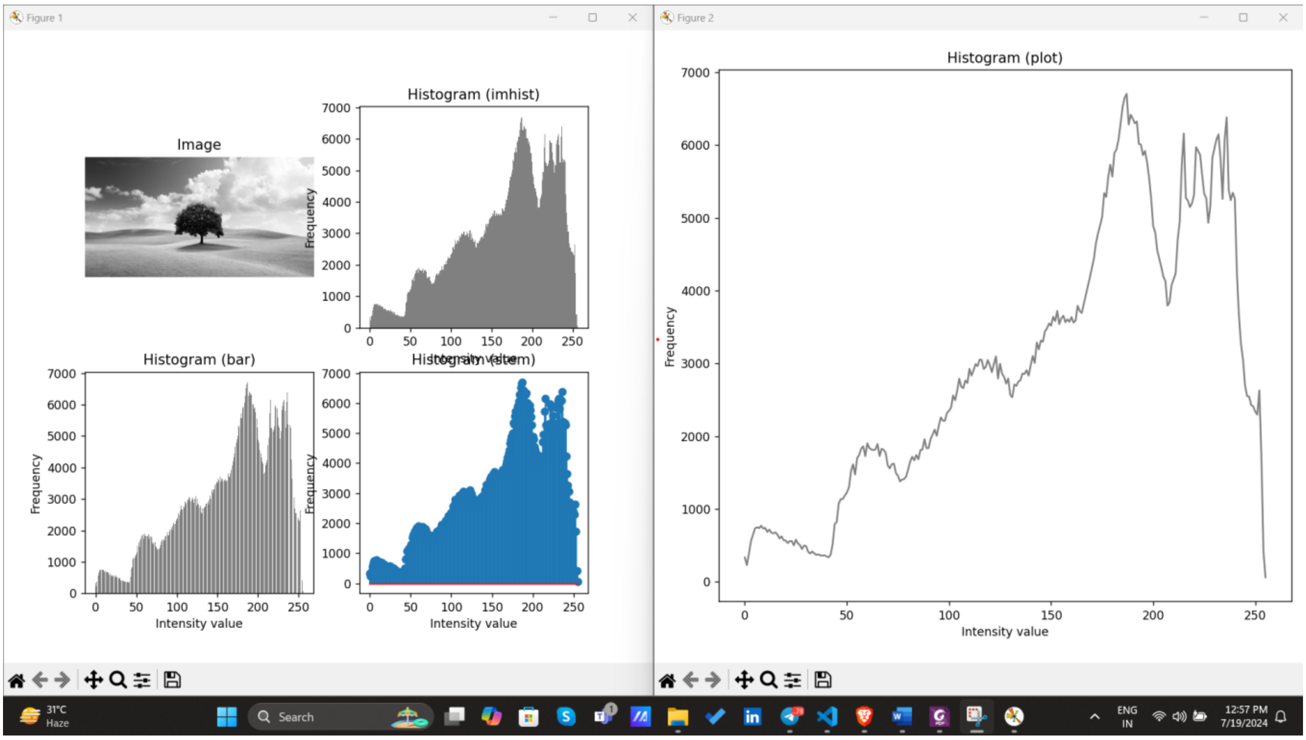
output:

8. Write a program to generate Histogram for an Image and plot histogram in variousways (imhist, bar, stem, plot).

Source Code:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt


def plot_histogram(image):

    hist = cv2.calcHist([image], [0], None, [256], [0, 256])


    hist_flat = hist.flatten()

    fig, axs = plt.subplots(2, 2, figsize=(10, 8))


    axs[0, 0].imshow(image, cmap='gray')

    axs[0, 0].set_title('Image')

    axs[0, 0].axis('off')


    axs[0, 1].hist(image.ravel(), bins=256, range=[0, 256], color='gray')

    axs[0, 1].set_title('Histogram (imhist)')

    axs[0, 1].set_xlabel('Intensity value')

    axs[0, 1].set_ylabel('Frequency')


    axs[1, 0].bar(np.arange(256), hist_flat, color='gray')

    axs[1, 0].set_title('Histogram (bar)')

    axs[1, 0].set_xlabel('Intensity value')

    axs[1, 0].set_ylabel('Frequency')


    axs[1, 1].stem(hist_flat)

    axs[1, 1].set_title('Histogram (stem)')

    axs[1, 1].set_xlabel('Intensity value')

    axs[1, 1].set_ylabel('Frequency')
    plt.figure(figsize=(8, 6))
    plt.plot(hist_flat, color='gray')
    plt.title('Histogram (plot)')
    plt.xlabel('Intensity value')
    plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()

image_path = "P:\\PCA2_10071023015\\nature.jpg"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

if image is None:
    print(f"Error: Unable to load image from {image_path}")
else:
    plot_histogram(image)
```

output:

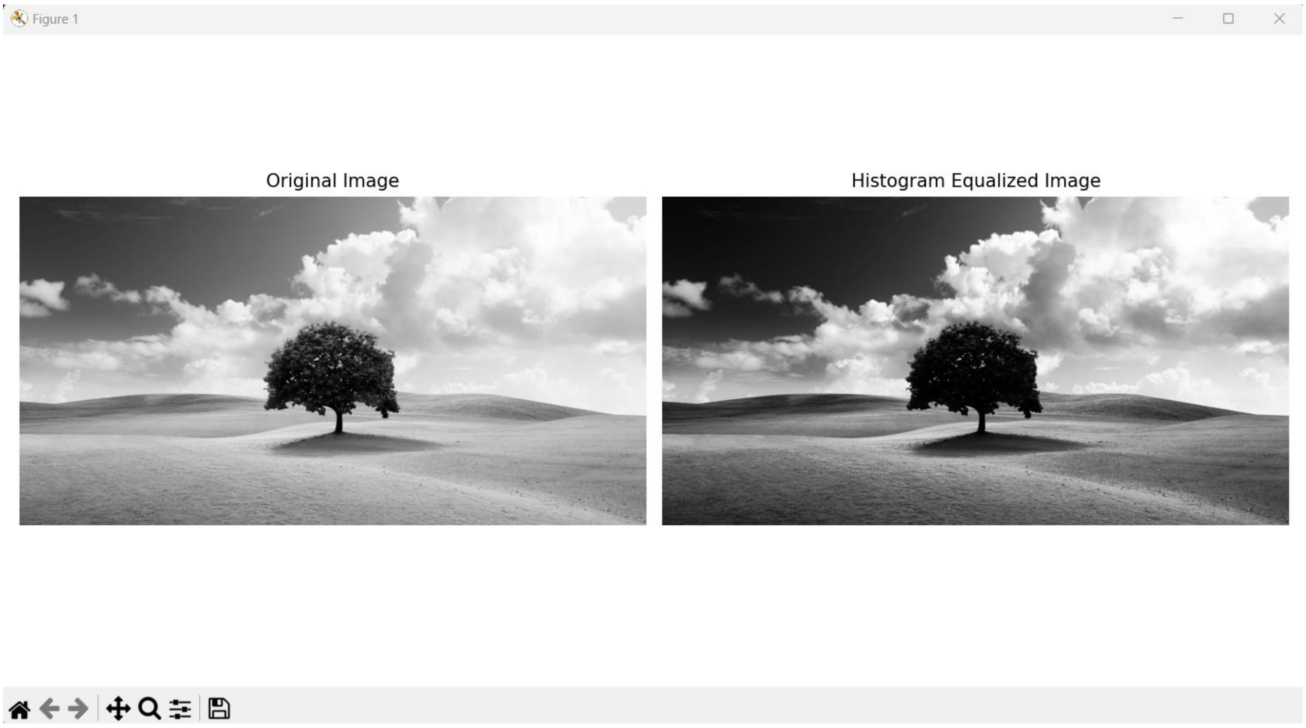


9. Write a program to perform Histogram Equalization

Source code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def histogram_equalization(image_path):
    # Load the image
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
     if image is None:
        print(f"Error: Unable to load image from {image_path}")
        return
     # Perform histogram equalization
    equalized_image = cv2.equalizeHist(image)
     # Plotting
    fig, axs = plt.subplots(1, 2, figsize=(12, 6))
    # Original Image
    axs[0].imshow(image, cmap='gray')
    axs[0].set_title('Original Image')
    axs[0].axis('off')
      # Equalized Image
    axs[1].imshow(equalized_image, cmap='gray')
    axs[1].set_title('Histogram Equalized Image')
    axs[1].axis('off')
    # Show plot
    plt.tight_layout()
    plt.show()
# Path to your image
image_path = "P:\\PCA2_10071023015\\nature.jpg"

# Perform histogram equalization
histogram_equalization(image_path)
```

output:



Original Image

Histogram Equalized Image

10. Write a program to implement Arithmetic and Logical operation

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

def image_subtraction(image1_path, image2_path):
    # Load images
    image1 = cv2.imread(image1_path)
    image2 = cv2.imread(image2_path)

    if image1 is None or image2 is None:
        print(f"Error: Unable to load images from {image1_path} or {image2_path}")
        return

    # Convert to grayscale
    gray1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

    # Check if dimensions match
    if gray1.shape != gray2.shape:
        # Resize gray1 to match gray2 dimensions
        gray1 = cv2.resize(gray1, (gray2.shape[1], gray2.shape[0]))

```

```python
    # Perform subtraction
    subtracted_image = cv2.subtract(gray1, gray2)

    # Plotting
    fig, axs = plt.subplots(1, 3, figsize=(15, 5))

    # Original Images
    axs[0].imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
    axs[0].set_title('Image 1')
    axs[0].axis('off')

    axs[1].imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
    axs[1].set_title('Image 2')
    axs[1].axis('off')

    # Subtracted Image
    axs[2].imshow(subtracted_image, cmap='gray')
    axs[2].set_title('Subtracted Image')
    axs[2].axis('off')

    # Show plot
    plt.tight_layout()
    plt.show()

# Paths to your images
image1_path = "P:\\PCA2_10071023015\\nature.jpg"  # Replace with your image path
image2_path = "P:\\PCA2_10071023015\\image.jpg"  # Replace with your image path

# Perform image subtraction
image_subtraction(image1_path, image2_path)

```

output:

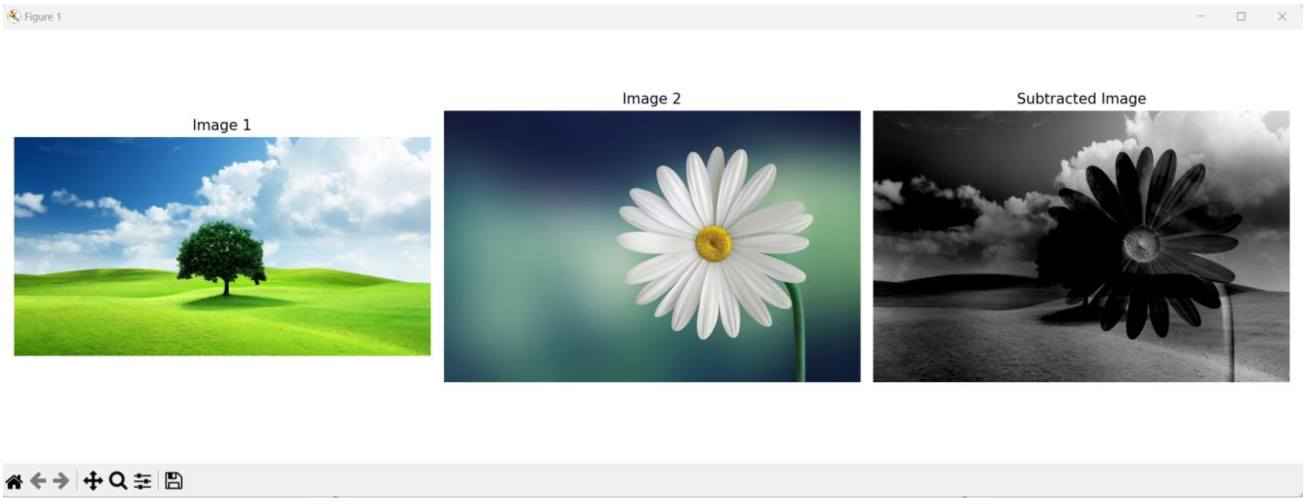


Figure 1
Image 1
Image 2
Subtracted Image