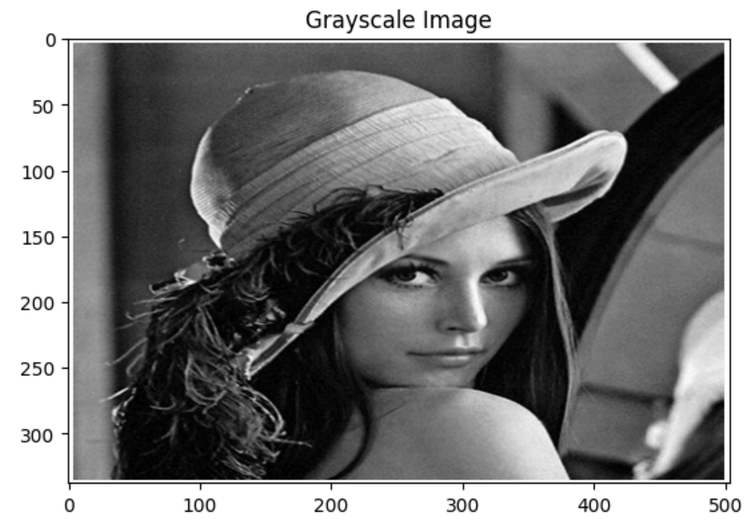
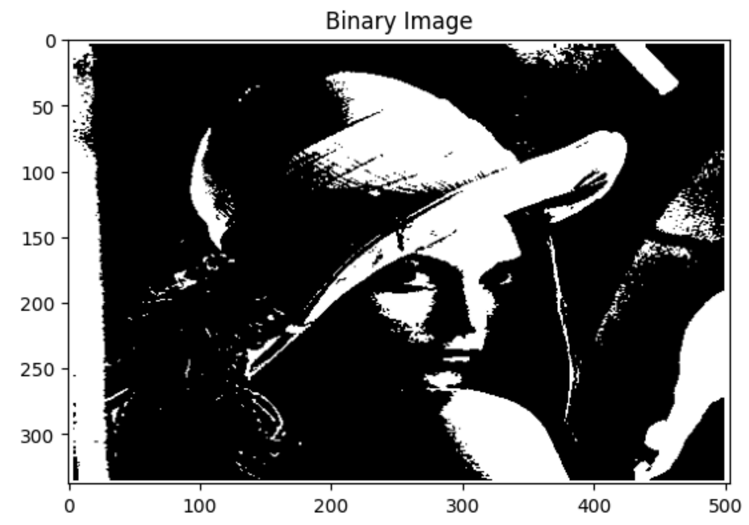
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Task 1: Read and Display Image
image = cv2.imread('/content/Screenshot 2025-01-30 112851.png')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.imshow(image_rgb)
plt.title('Original Image')
plt.show()

# Extract Image Size
height, width, channels = image.shape
print(f'Image Size: {width}x{height}, Channels: {channels}')

# Calculate Image Pixels
total_pixels = height * width
print(f'Total Pixels: {total_pixels}')

# Convert RGB to Grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imwrite('gray_image.jpg', gray_image)
plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale Image')
plt.show()

# Convert to Binary Image using a threshold
_, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
cv2.imwrite('binary_image.jpg', binary_image)
plt.imshow(binary_image, cmap='gray')
plt.title('Binary Image')
plt.show()

# Count Black Pixels
black_pixel_count = np.sum(binary_image == 0)
print(f'Number of Black Pixels: {black_pixel_count}')

# Task 2: Edge Detection
# Sobel Operator
sobel_x = cv2.Sobel(gray_image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(gray_image, cv2.CV_64F, 0, 1, ksize=3)
sobel_combined = cv2.magnitude(sobel_x, sobel_y)
plt.imshow(sobel_combined, cmap='gray')
plt.title('Sobel Edge Detection')
plt.show()

# Prewitt Operator
kernelx = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
kernely = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
prewitt_x = cv2.filter2D(gray_image, cv2.CV_64F, kernelx) # Changed -1 to cv2.CV_64F
prewitt_y = cv2.filter2D(gray_image, cv2.CV_64F, kernely) # Changed -1 to cv2.CV_64F
prewitt_combined = cv2.magnitude(prewitt_x, prewitt_y)
plt.imshow(prewitt_combined, cmap='gray')
plt.title('Prewitt Edge Detection')
plt.show()

# Roberts Cross Operator
roberts_x = np.array([[1, 0], [0, -1]])
roberts_y = np.array([[0, 1], [-1, 0]])
roberts_x = cv2.filter2D(gray_image, cv2.CV_64F, roberts_x) # Changed -1 to cv2.CV_64F
roberts_y = cv2.filter2D(gray_image, cv2.CV_64F, roberts_y) # Changed -1 to cv2.CV_64F
roberts_combined = cv2.magnitude(roberts_x, roberts_y)
plt.imshow(roberts_combined, cmap='gray')
plt.title('Roberts Cross Edge Detection')
plt.show()
# Canny Edge Detection
canny_edges = cv2.Canny(gray_image, 100, 200)
plt.imshow(canny_edges, cmap='gray')
plt.title('Canny Edge Detection')
plt.show()

# Task 2: Image Segmentation
# Global Thresholding
_, global_thresh = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
plt.imshow(global_thresh, cmap='gray')
plt.title('Global Thresholding')
plt.show()

# Adaptive Thresholding
adaptive_thresh = cv2.adaptiveThreshold(gray_image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
```

```
    plt.imshow(adaptive_thresh, cmap='gray')
    plt.title('Adaptive Thresholding')
    plt.show()

    # Edge Detection for Segmentation (Canny)
    plt.imshow(canny_edges, cmap='gray')
    plt.title('Edge Detection for Segmentation')
    plt.show()

    # Region-Based Segmentation (Watershed Algorithm)
    gray_blurred = cv2.GaussianBlur(gray_image, (5, 5), 0)
    _, binary_thresh = cv2.threshold(gray_blurred, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    # Distance transform and marker labeling
    dist_transform = cv2.distanceTransform(binary_thresh, cv2.DIST_L2, 5)
    _, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)
    sure_fg = np.uint8(sure_fg)
    unknown = cv2.subtract(binary_thresh, sure_fg)

    # Marker labelling
    _, markers = cv2.connectedComponents(sure_fg)
    markers += 1
    markers[unknown == 255] = 0
    markers = cv2.watershed(image, markers)
    image[markers == -1] = [255, 0, 0]
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title('Watershed Segmentation')
    plt.show()
```

```
Image Size: 504x338, Channels: 3
Total Pixels: 170352
```





```
Number of Black Pixels: 125168
```