```
NAME = "Loknath Saha"
ID = "2020107"
```

Necessary library import

```
import numpy as np
from skimage import io, color, exposure, img_as_float
import matplotlib.pyplot as plt
```

Task 1 - Basic Image Operation

import your image or any photo taken by you (`sample.jpeg`) as a numpy array, save it in the varaible `I`.

A picture taken from your phone of any scenary/streets/building is better.

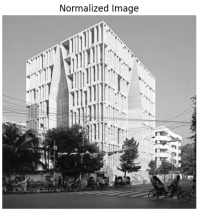
remember your image name MUST be `sample.jpeg`.

Make sure the height and the width of the image is **smaller than 1000 pixels**.

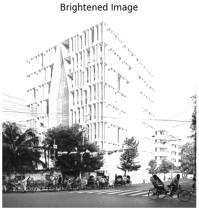
```
I = io.imread("sample.jpeg") # Replace None with appropriate function call line


# find the height and the width of the image

H,W = I.shape[:2]

print("Height is", H)
print("Width is", W)
```

```
Height is 750
Width is 750
```

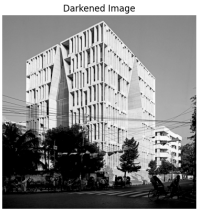
```
# Normalize the image so that the gray scales are between 0 and 1. Save it to I and display the image
I = io.imread("sample.jpeg")

### BEGIN SOLUTION
I2 = color.rgb2gray(I)
#print(I2.shape)
I3 = img_as_float(I2)
#or I3 = I2/255.0
plt.imshow(I3, cmap='gray')
plt.title('Normalized Image')
plt.axis('off')
plt.show()
### END SOLUTION
```




```
# Increase the brightness of the image without changing the contrast.
# Save the resulting image in I_bright and display it.
I_bright = np.clip(I3+0.4,0,1)
plt.imshow(I_bright, cmap='gray')
plt.title('Brightened Image')
plt.axis('off')
plt.show()
```



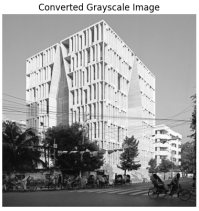

```
# Decrease the brightness of the image without changing the contrast.
# Save the resulting image in I_dark and display it.
I_dark = np.clip(I3-0.2, 0, 1)

plt.imshow(I_dark, cmap='gray')
plt.title('Darkened Image')
plt.axis('off')
plt.show()

#io.imsave('dark_image.jpg', (I_dark * 255).astype(np.uint8))
```



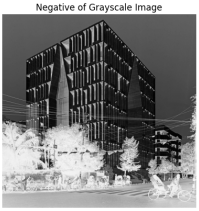

```
# Multiply the three channels of image I with three DIFFERENT numbers between 0.3 and
# Save the resulting image in I_tint and display it.
```

```
# The resulting image should have some color shift
#I_tint = None


# HINT:
# I_tint = np.zeros(I.shape)
# I_tint[:, :, 0] =  ..... I[:, :, 0].....
# ......

### BEGIN SOLUTION
I_tint = np.zeros(I.shape,dtype=np.float64)
scale_r = 1.5
scale_g = 0.7
scale_b = 2.0


I_tint[:, :, 0] = I[:, :, 0] * scale_r
I_tint[:, :, 1] = I[:, :, 1] * scale_g
I_tint[:, :, 2] = I[:, :, 2] * scale_b

I_tint = np.clip(I_tint, 0, 255).astype(np.uint8)

plt.imshow(I_tint)
plt.title('Tinted Image with Color Shift')
plt.axis('off')
plt.show()

### END SOLUTION
```


```
# Convert the image into a grayscale image.
# Save it to I_gray and display it
# Gray=0.2126×R+0.7152×G+0.0722×B
#I_gray = None

### BEGIN SOLUTION

#or I_gray = 0.2126 * I[:, :, 0] + 0.7152 * I[:, :, 1] + 0.0722 * I[:, :, 2]
I_gray = color.rgb2gray(I)
I_gray = img_as_float(I_gray)


plt.imshow(I_gray, cmap='gray')
plt.title('Converted Grayscale Image')
plt.axis('off')
plt.show()

### END SOLUTION
```




```
# Display the negative of the grayscale image

### BEGIN SOLUTION
I_negative = 1 - I_gray
plt.imshow(I_negative, cmap='gray')
plt.title('Negative of Grayscale Image')
plt.axis('off')
plt.show()
### END SOLUTION
```




```
# Artificially degrade the **grayscale image** by reducing it contrast
# You can do so by recaling the gray values and concentrating them in a narrow range
# say between 0.3 and 0.6.
# Save the image as I_degraded and display it
# HINT: SEE lec-4-demo-codes

new_min, new_max = 0.4, 0.5
I10 = img_as_float(I_gray)

I_degraded = (I10 * (new_max - new_min)) + new_min
I_degraded = np.clip(I_degraded, 0, 1)


plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(I_gray, cmap='gray')
plt.title('Original Image')
plt.axis('off')
```

```
plt.subplot(1, 2, 2)
plt.imshow(I_degraded, cmap='gray')
plt.title('Degraded Contrast Image')
plt.axis('off')

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

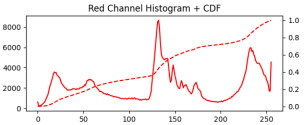

```
# Complete the following function to perform Piecewise Linear Contrast stretching
# That is, implement the map shown in Slide 17 of Lecture 3

# Prototype: piecewise_contrast_stretch(I_gray, r1, r2, s1, s2)
# Assuming both input and output images are normalized between 0 and 1

def piecewise_contrast_stretch(I, r1, r2, s1, s2):
    pass
    # Write your code here
    I_stretched = np.zeros_like(I)

    # Apply the first mapping [0, r1] -> [0, s1]
    mask1 = I <= r1
    I_stretched[mask1] = (s1 / r1) * I[mask1]

    # Apply the second mapping [r2, 1] -> [s2, 1]
    mask2 = I > r2
    I_stretched[mask2] = ((1 - s2) / (1 - r2)) * (I[mask2] - r2) + s2

    # Apply the third mapping [r1, r2] -> [s1, s2]
    mask3 = (I > r1) & (I <= r2)
    I_stretched[mask3] = ((s2 - s1) / (r2 - r1)) * (I[mask3] - r1) + s1

    return I_stretched
```

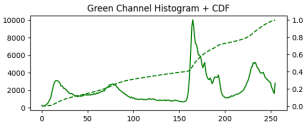
```
# To test your implementation, contrast strech the degraded image I_degrade
r1 = 0.2
r2 = 0.7
s1 = 0.1
s2 = 0.9
I_stretched = piecewise_contrast_stretch(I_degraded, r1, r2, s1, s2)

plt.imshow(I_stretched, cmap='gray')
plt.title('Piecewise Linear Contrast Stretched Image')
plt.axis('off')
plt.show()
```

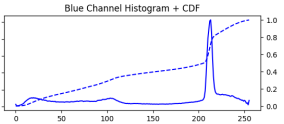
Task 2 - Histogram and Equalization

```
# Plot the Image and its histogram + cdf of the original image I
# Note that it is a color image, so it will have three different histograms

### BEGIN SOLUTION
I = io.imread("sample.jpeg")


R = I[:, :, 0]
G = I[:, :, 1]
B = I[:, :, 2]

def plot_histogram_and_cdf(channel, color_name):

    hist, bins = exposure.histogram(channel, nbins=256, normalize=False)

    cdf, cdf_bins = exposure.cumulative_distribution(channel, 256)

    plt.plot(bins, hist, label=f'{color_name} Histogram', color=color_name)


    plt.twinx()
    plt.plot(cdf_bins, cdf, label=f'{color_name} CDF', color=color_name, linestyle='--')


plt.figure(figsize=(12, 5))
plt.subplot(2, 2, 1)
plt.imshow(I)
plt.title('Original Image')
plt.axis('off')

plt.subplot(2, 2, 2)
plot_histogram_and_cdf(R, 'red')
plt.title('Red Channel Histogram + CDF')


plt.subplot(2, 2, 3)
plot_histogram_and_cdf(G, 'green')
plt.title('Green Channel Histogram + CDF')
```

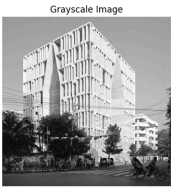
```
plt.subplot(2, 2, 4)
plot_histogram_and_cdf(B, 'blue')
plt.title('Blue Channel Histogram + CDF')

plt.tight_layout()
plt.show()

### END SOLUTION
```

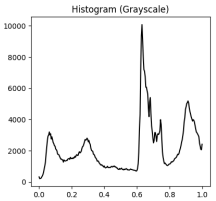


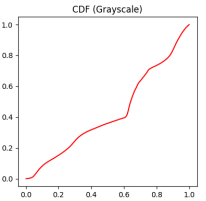




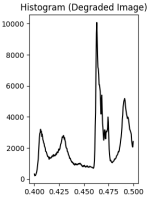

```
# Plot the Image and its histogram + cdf of the grayscale image I_gray

### BEGIN SOLUTION

hist, bins = exposure.histogram(I_gray, nbins=256, normalize=False)
cdf, cdf_bins = exposure.cumulative_distribution(I_gray, 256)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.imshow(I_gray, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.plot(bins, hist, color='black')
plt.title('Histogram (Grayscale)')

plt.subplot(1, 3, 3)
plt.plot(cdf_bins, cdf, color='red')
plt.title('CDF (Grayscale)')

plt.tight_layout()
plt.show()
### END SOLUTION
```

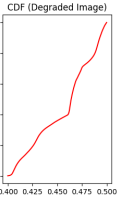






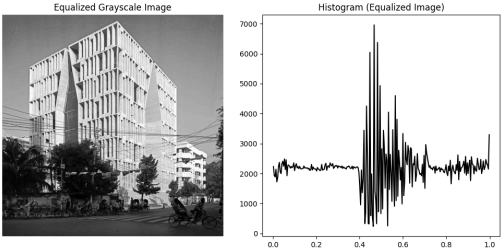

```
# Plot the Image and its histogram + cdf of the degraded image I_degraded

### BEGIN SOLUTION
hist, bins = exposure.histogram(I_degraded, nbins=256, normalize=False)
cdf, cdf_bins = exposure.cumulative_distribution(I_degraded, 256)


plt.figure(figsize=(8,4))

plt.subplot(1, 3, 1)
plt.imshow(I_degraded, cmap='gray')
plt.title('Degraded Grayscale Image')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.plot(bins, hist, color='black')
plt.title('Histogram (Degraded Image)')

plt.subplot(1, 3, 3)
plt.plot(cdf_bins, cdf, color='red')
plt.title('CDF (Degraded Image)')

plt.tight_layout()
plt.show()
### END SOLUTION
```



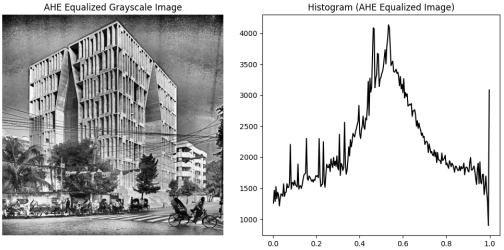



```
# Equalize the histogram of the degraded image I_degraded
# Save the result in I_recon_gray, display the image along with its histogram

I_recon_gray = exposure.equalize_hist(I_degraded)
hist_recon, bins_recon = exposure.histogram(I_recon_gray, nbins=256, normalize=False

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_recon_gray, cmap='gray')
```

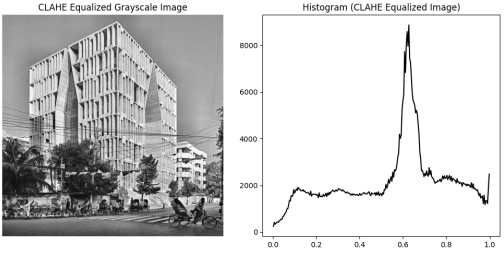
```
plt.title('Equalized Grayscale Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.plot(bins_recon, hist_recon, color='black')
plt.title('Histogram (Equalized Image)')

plt.tight_layout()
```




```
# Equalize the histogram of the degraded image I_degraded using AHE
# Save the result in I_recon_gray_2, display the image along with its histogram

I_recon_gray_2 = exposure.equalize_adapthist(I_degraded,kernel_size=(128, 128),clip_limit=0)

hist_recon_2, bins_recon_2 = exposure.histogram(I_recon_gray_2, nbins=256, normalize=False)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_recon_gray_2, cmap='gray')
plt.title('AHE Equalized Grayscale Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.plot(bins_recon_2, hist_recon_2, color='black')
plt.title('Histogram (AHE Equalized Image)')

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

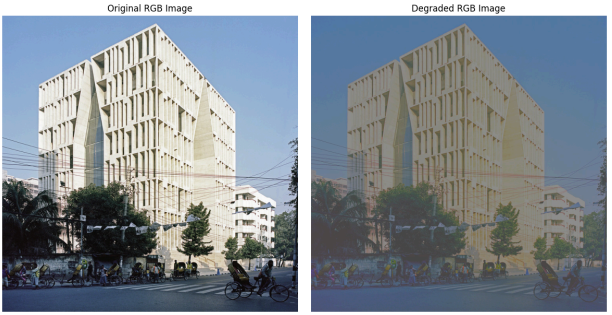



```
# Equalize the histogram of the degraded image I_degraded using CLAHE
# Save the result in I_recon_gray_2, display the image along with its histogram


I_recon_gray_3 = exposure.equalize_adapthist(I_degraded,kernel_size=(64, 64), clip_limit=0.01)

hist_recon_3, bins_recon_3 = exposure.histogram(I_recon_gray_3, nbins=256, normalize=False)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_recon_gray_3, cmap='gray')
plt.title('CLAHE Equalized Grayscale Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.plot(bins_recon_3, hist_recon_3, color='black')
plt.title('Histogram (CLAHE Equalized Image)')

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



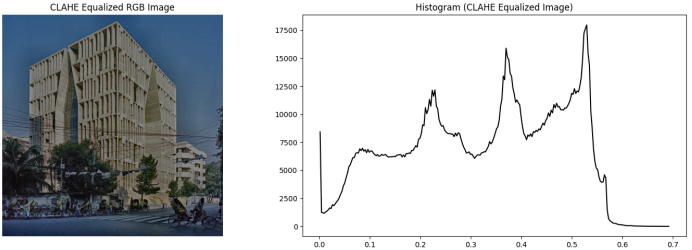

```
# Artificially degrade the original **RGB image** by reducing it contrast
# You can do so by recaling the values of the L channel (in LAB color space)
# and concentrating them in a narrow range, say between 0.3 and 0.6.
# Save the image as I_rgb_degraded and display it


### BEGIN SOLUTION
I_lab = color.rgb2lab(I)
L = I_lab[:, :, 0]

```

```
new_min, new_max = 0.3 * 100, 0.6 * 100

L_norm = (L - L.min()) / (L.max() - L.min())

L_degraded = L_norm * (new_max - new_min) + new_min

I_lab_degraded = I_lab.copy()
I_lab_degraded[:, :, 0] = L_degraded

I_rgb_degraded = color.lab2rgb(I_lab_degraded)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(I)
plt.title('Original RGB Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(I_rgb_degraded)
plt.title('Degraded RGB Image')
plt.axis('off')

plt.tight_layout()
plt.show()

print(f"Original L channel range: [{L.min():.2f}, {L.max():.2f}]")
print(f"Degraded L channel range: [{L_degraded.min():.2f}, {L_degraded.max():.2f}]"
### END SOLUTION
```




```
Original L channel range: [0.00, 100.00]
Degraded L channel range: [30.00, 60.00]
```

```
# Equalize the histogram of the degraded color image I_rgb_degraded using CLAHE
# Save the result in I_recon_color, display the image along with its histogram
# HINT: You have to convert to LAB first
# See the lecture and lecture-4-demo-codes
I_lab_degraded = color.rgb2lab(I_rgb_degraded)

m12 = I_lab_degraded[:,:,0].min()
m2 = I_lab_degraded[:,:,0].max()

L_degraded  = I_lab_degraded[:, :, 0] /m2

L_degraded_clahe = exposure.equalize_adapthist(L_degraded,kernel_size=(64, 64), clip_limit=0.01)

I_lab_recon = I_lab_degraded.copy()
I_lab_recon[:, :, 0] = L_degraded_clahe * m2


I_recon_color = color.lab2rgb(I_lab_recon)


hist_recon_color, bins_recon_color = exposure.histogram(I_recon_color, nbins=256, normalize=False)


plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_recon_color)
plt.title('CLAHE Equalized RGB Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.plot(bins_recon_color, hist_recon_color, color='black')
plt.title('Histogram (CLAHE Equalized Image)')

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



Task 3 - Open Ended

There are 3 images in the drive directory below. Look at the questions from the brackets [.].
Answer them in the provided text cell at the bottom.

link: https://drive.google.com/drive/folders/1ft3XrO-MGxhxL2PfcLCFOjU0wvvILAv4?usp=sharing

```
# Dark_Room.jpg = very dark [The windows are on walls. How does the wall look like?]
# Foggy_Road.jpg = washed out/foggy [How many vehicles do you think there are?]
```

```
# Read_the_code.jpg = Dark RGB Barcode [What is hidden in the Barcode?
#                                       Make it scanable, scan it and say something about the hidden message.

# Your task is to improve these images using
# contrast stretching, histogram equalization, AHE or CLAHE
# try different combination of parameter settings to see which produces the best result

### BEGIN SOLUTION
```

Your answers:

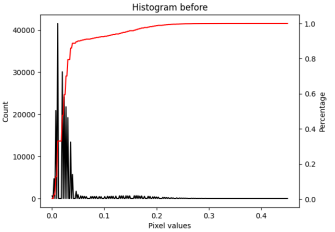
abc abc abc ... ...

```
I_dark_room = io.imread('Dark_Room.jpg')
I_dark_room.shape
```

```
(342, 600, 3)
```

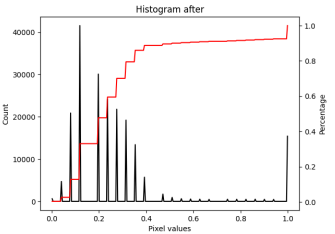
```
I_dark_room_gray = color.rgb2gray(I_dark_room)
```

```
hist, bins_hist = exposure.histogram(I_dark_room_gray, nbins=256, normalize=False)
plt.plot(bins_hist, hist, 'k')
plt.xlabel("Pixel values")
plt.ylabel("Count")
plt.title("Histogram before")

plt.twinx()
img_cdf, bins = exposure.cumulative_distribution(I_dark_room_gray, 256)
plt.plot(bins, img_cdf, "red")
plt.ylabel("Percentage")
plt.show()
```




```
I_stretched = np.clip(10*(I_dark_room_gray), 0, 1)
```

```
hist, bins_hist = exposure.histogram(I_stretched, nbins=256, normalize=False)
plt.plot(bins_hist, hist, 'k')
plt.xlabel("Pixel values")
plt.ylabel("Count")
plt.title("Histogram after")

plt.twinx()
img_cdf, bins = exposure.cumulative_distribution(I_stretched, 256)
plt.plot(bins, img_cdf, "red")
plt.ylabel("Percentage")
plt.show()
```

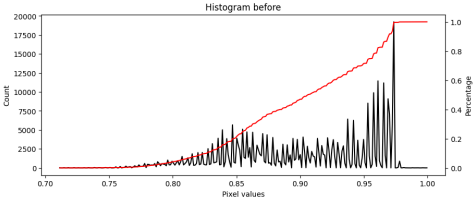



```
plt.figure(figsize=(8, 4))
io.imshow(I_stretched)
plt.axis("off")
plt.show()
```

```
I_dark_room_clahe = exposure.equalize_adapthist(I_dark_room_gray,kernel_size=(64, 64), clip_limit=0.03


hist, bins_hist = exposure.histogram(I_dark_room_clahe , nbins=256, normalize=False)

plt.figure(figsize=(10, 4))
plt.plot(bins_hist, hist, 'k')
plt.xlabel("Pixel values")
plt.ylabel("Count")
plt.title("Histogram before")

plt.twinx()
img_cdf, bins = exposure.cumulative_distribution(I, 256)
plt.plot(bins, img_cdf, "red")
plt.ylabel("Percentage")
plt.show()
```

```
plt.figure(figsize=(10, 10))

plt.subplot(1, 2, 1)
plt.imshow(I_dark_room)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(I_dark_room_clahe, cmap='gray')
plt.title('CLAHE')
plt.axis('off')

plt.show()
```


```
I_foggy_road = io.imread('Foggy_Road.jpg')

I_foggy_road_gray = color.rgb2gray(I_foggy_road)


hist, bins_hist = exposure.histogram(I_foggy_road_gray, nbins=256, normalize=False)

plt.figure(figsize=(10, 4))
plt.plot(bins_hist, hist, 'k')
plt.xlabel("Pixel values")
plt.ylabel("Count")
plt.title("Histogram before")

plt.twinx()
img_cdf, bins = exposure.cumulative_distribution(I_foggy_road_gray, 256)
plt.plot(bins, img_cdf, "red")
plt.ylabel("Percentage")
plt.show()

```

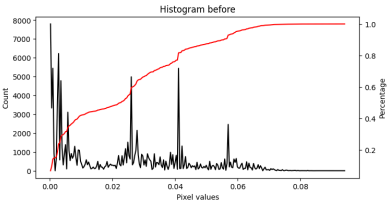



```
# Apply Histogram Equalization
Ieq = exposure.equalize_hist(I_foggy_road_gray)
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_foggy_road)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 1)
plt.imshow(I_foggy_road)
plt.title('Original Image')
plt.axis('off')


plt.subplot(1, 2, 2)
plt.imshow(Ieq, cmap='gray')
plt.title('Histogram Equalization')
plt.axis('off')

plt.show()
```

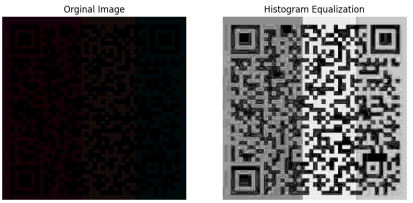

```
# Apply CLAHE
I_foggy_road_clahe = exposure.equalize_adapthist(I_foggy_road_gray,kernel_size=(64, 64), clip_limit=0.01


plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_foggy_road)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(I_foggy_road_clahe, cmap='gray')
plt.title('CLAHE')
plt.axis('off')
```

```
16
```


```
I_read_code = io.imread('Read_the_code.jpg')

I_read_code_gray = color.rgb2gray(I_read_code)


hist, bins_hist = exposure.histogram(I_read_code_gray, nbins=256, normalize=False)
plt.figure(figsize=(8, 4))
plt.plot(bins_hist, hist, 'k')
plt.xlabel("Pixel values")
plt.ylabel("Count")
plt.title("Histogram before")

plt.twinx()
img_cdf, bins = exposure.cumulative_distribution(I_read_code_gray, 256)
plt.plot(bins, img_cdf, "red")
plt.ylabel("Percentage")
plt.show()

```




```

Ieq = exposure.equalize_hist(I_read_code_gray)
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_read_code)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 1)
plt.imshow(I_read_code)
plt.title('Orginal Image')
plt.axis('off')


plt.subplot(1, 2, 2)
plt.imshow(Ieq, cmap='gray')
plt.title('Histogram Equalization')
plt.axis('off')

plt.show()
```




```
I_read_code_clahe = exposure.equalize_adapthist(I_read_code_gray,kernel_size=(40, 40), clip_limit=0.02)


plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(I_read_code)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(I_read_code_clahe, cmap='gray')
plt.title('CLAHE')
plt.axis('off')

plt.show()
```