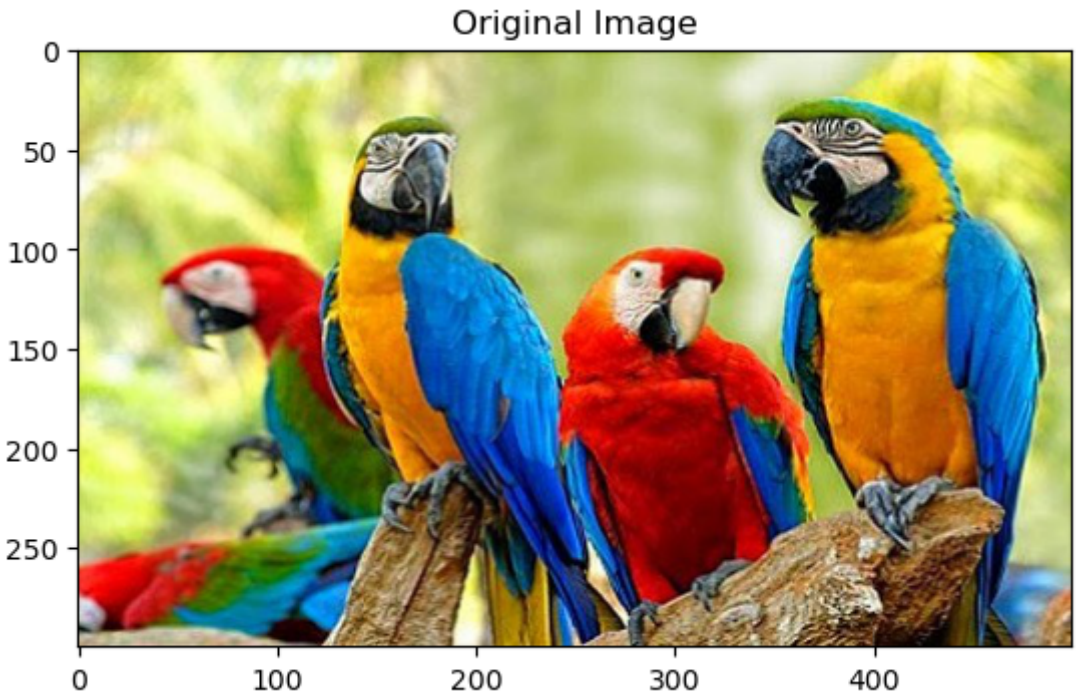
Assignment 1

Problem 1

In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
#Load Image as RBG
img = cv2.imread("bird2.jpg") #By Default it loads as BGR
img_rgb = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)   #converts color space from BGR
print(img_rgb.shape) #(300, 500, 3)

plt.figure()

#Display RGB Image
plt.title("Original Image")
plt.imshow(img_rgb)
```

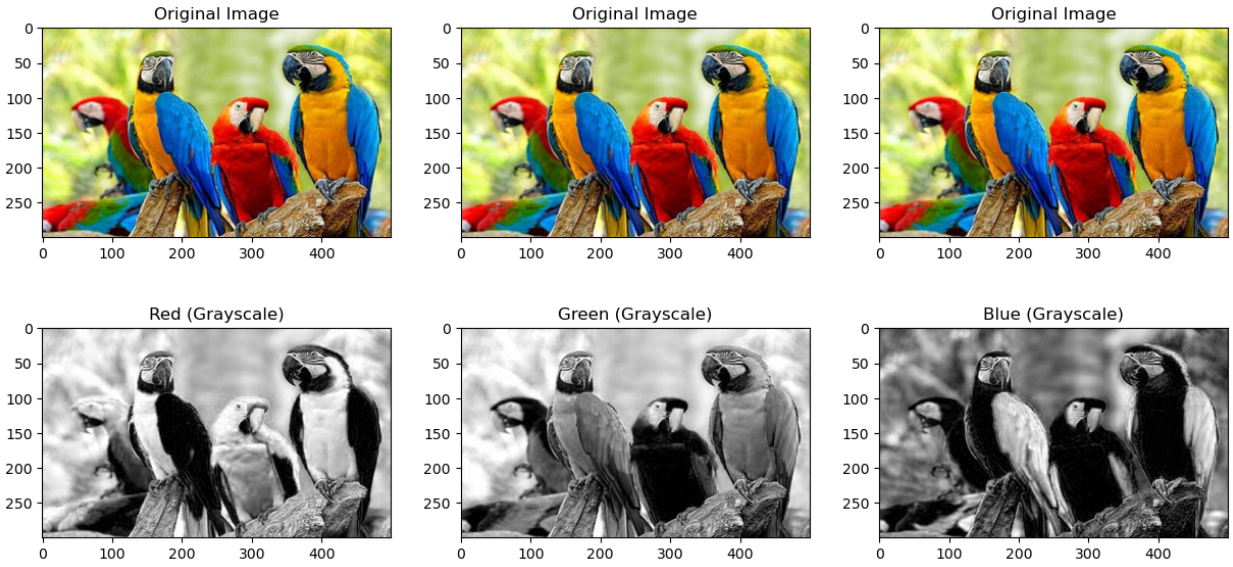
Out[ ]:
```
(300, 500, 3)
<matplotlib.image.AxesImage at 0x7fac70feb580>
```



In [ ]:
```
#Red Channel
img_r = img_rgb[:,:,0]

#Green Channel
img_g = img_rgb[:,:,1]

#Blue Channel
img_b = img_rgb[:,:,2]
```

```
plt.figure(figsize = (15,7))

plt.subplot(2,3,1)
plt.title('Original Image')
plt.imshow(img_rgb)
plt.subplot(2,3,2)
plt.title('Original Image')
plt.imshow(img_rgb)
plt.subplot(2,3,3)
plt.title('Original Image')
plt.imshow(img_rgb)
plt.subplot(2,3,4)
plt.title('Red (Grayscale)')
plt.imshow(img_r, cmap = 'gray')
plt.subplot(2,3,5)
plt.title('Green (Grayscale)')
plt.imshow(img_g, cmap = 'gray')
plt.subplot(2,3,6)
plt.title('Blue (Grayscale)')
plt.imshow(img_b, cmap = 'gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7facb4a1b9a0>
```

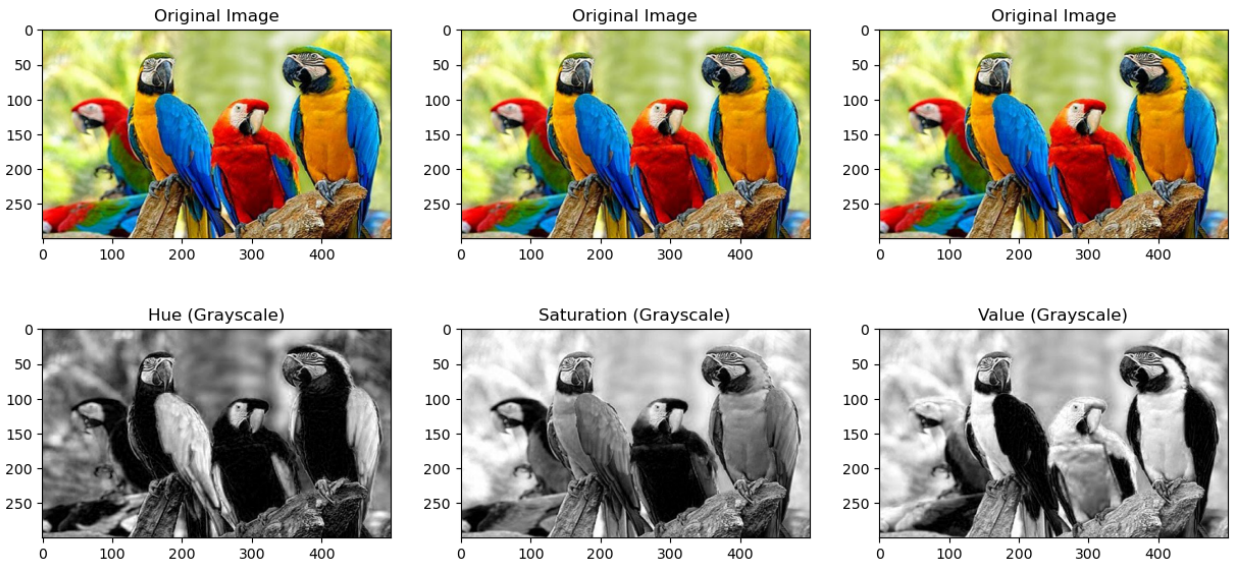


We can observe that while displaying single channel at time, part of the image which contains same color as channel color is more bright than others.

In [ ]:
```
#BGR to #HSV
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

plt.figure(figsize = (15,7))

plt.subplot(2,3,1)
plt.title('Original Image')
plt.imshow(img_rgb)
plt.subplot(2,3,2)
plt.title('Original Image')
plt.imshow(img_rgb)
plt.subplot(2,3,3)
plt.title('Original Image')
plt.imshow(img_rgb)
```

```
#Display H channel
plt.subplot(2,3,4)
plt.title('Hue (Grayscale)')
plt.imshow(img[:,:,0], cmap = 'gray')

#Display S channel
plt.subplot(2,3,5)
plt.title('Saturation (Grayscale)')
plt.imshow(img[:,:,1], cmap = 'gray')

#Display V channel
plt.subplot(2,3,6)
plt.title('Value (Grayscale)')
plt.imshow(img[:,:,2], cmap = 'gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7facb4a899d0>
```



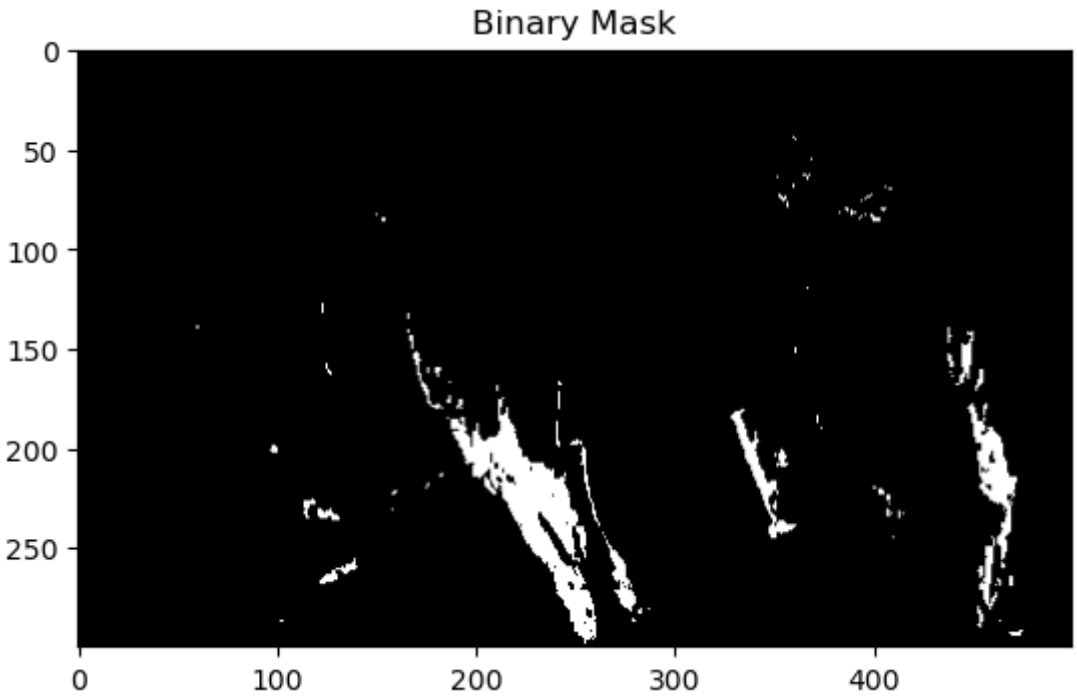
Hue Channel just shows different dominant colors

Saturation Channel dominance of particular hue color or we can say amount of white light to be mixed with hue color. For ex, red portion is black which means no or little amount of white is needed.

Value(Intensity or Brightness) shows lightness or darkness of particular color. For ex, blue portion is seen as black means shade of blue in image is darker. Likewise yellow portion is very bright.

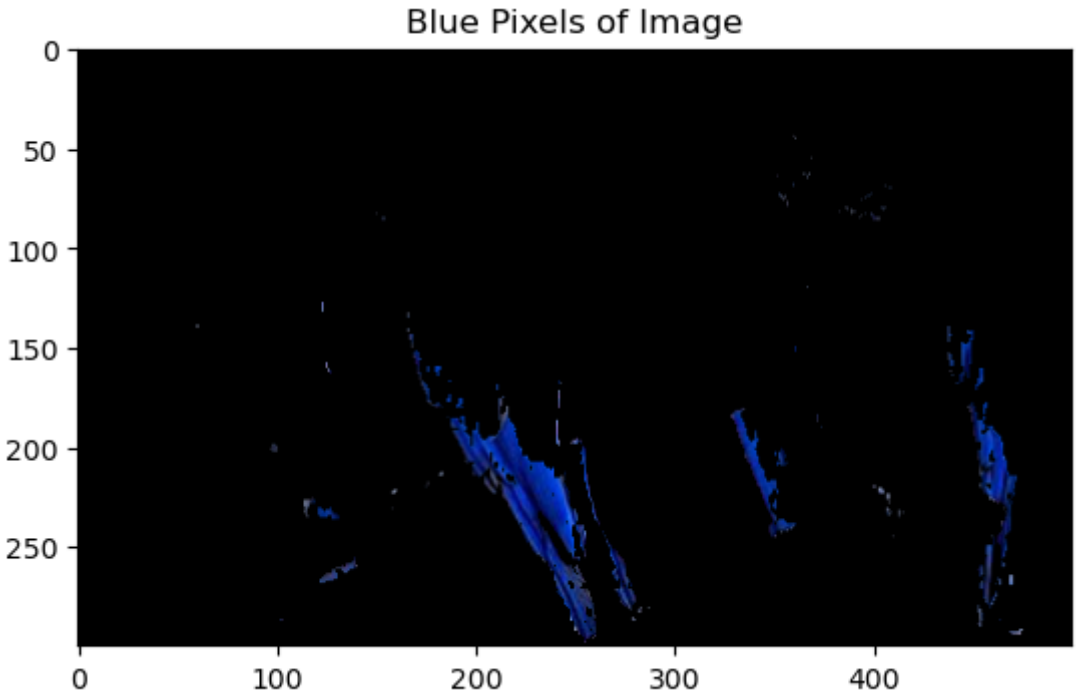
In [ ]:
```
# Range of Blue Hue from 110 to 130
blue_low = np.array([110, 50, 50])
blue_high = np.array([130, 255, 255])

binaryMask = cv2.inRange(img_hsv, blue_low, blue_high)    #it will have value as
blueOnly = cv2.bitwise_and(img, img, mask=binaryMask)     #It will apply mask to

plt.title("Binary Mask")
plt.imshow(binaryMask, cmap = 'gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fac50c63c70>
```



In [ ]:
```
#Displaying Blue pixels only
blueOnly_rgb = cv2.cvtColor(blueOnly, cv2.COLOR_BGR2RGB)
plt.title("Blue Pixels of Image")
plt.imshow(blueOnly_rgb)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fac719b4340>
```

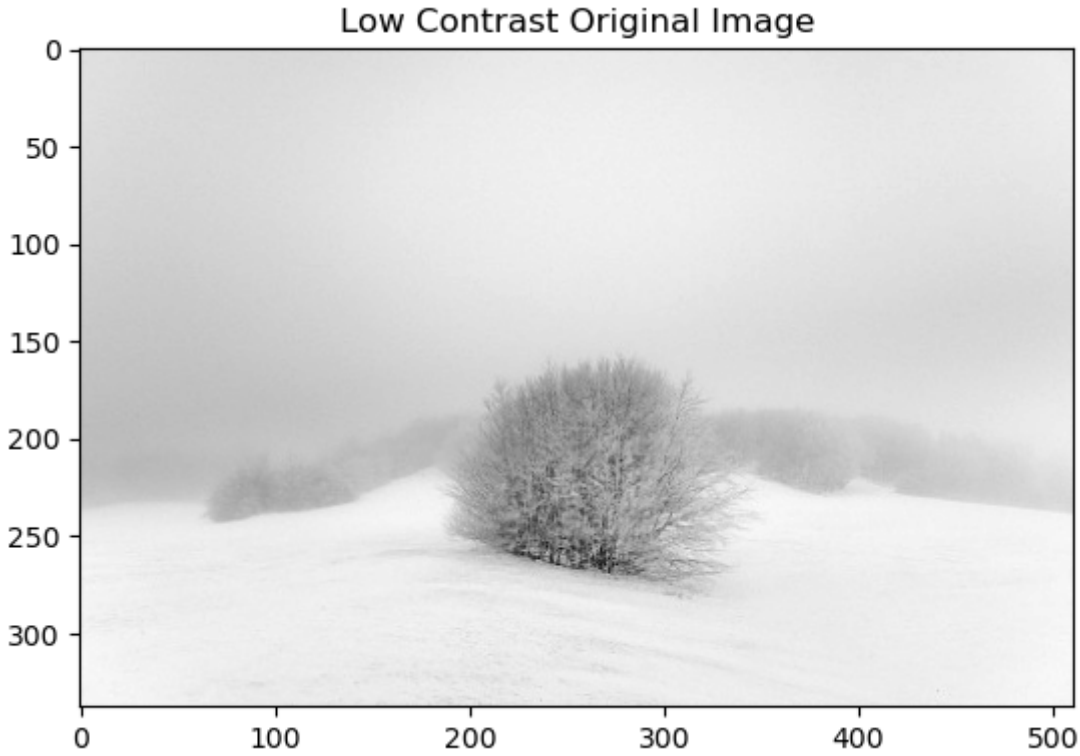


Problem 2

In [ ]:
```
#Load Image
img = cv2.imread("low4.jpeg")  #By Default it loads as BGR
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)   #converts color space from BG
```

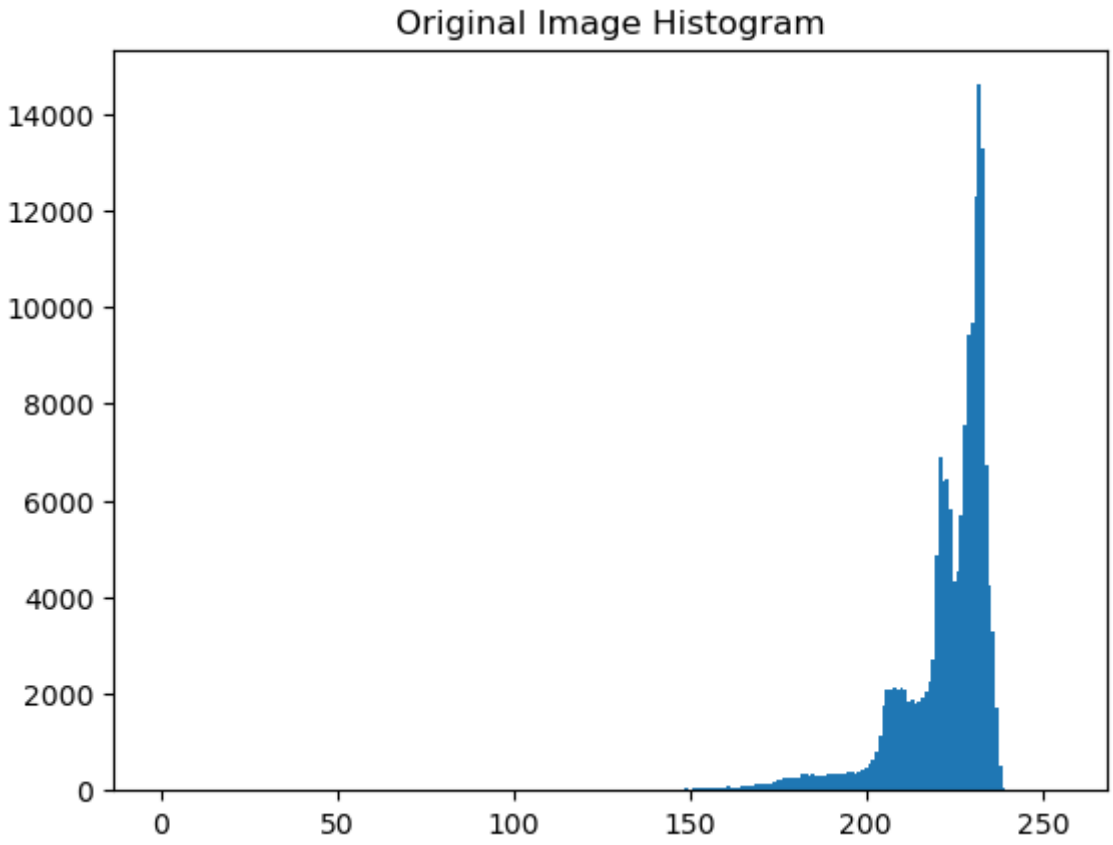
```
plt.title("Low Contrast Original Image")
plt.imshow(img_gray, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7facb4b54be0>
```



In [ ]:
```
#Histogram Calculation
histogram = [0]*256
print(len(histogram))
for rows in img_gray:
    for val in rows:
        histogram[val] = histogram[val] + 1

#Plot Histogram
idx = np.arange(256)
fig, ax = plt.subplots()
ax.bar(idx, histogram, 1)
plt.title("Original Image Histogram")
plt.show()
```

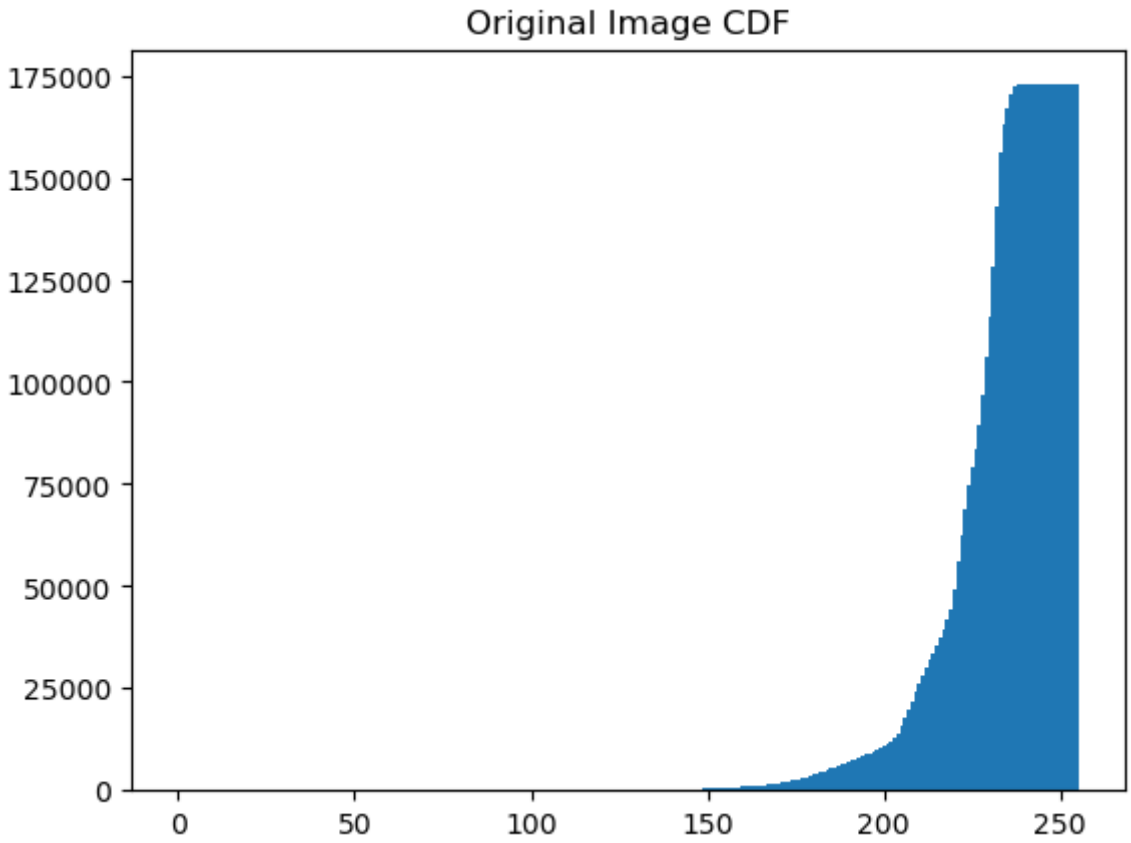
```
256
```



As we can see from histogram that values from upper range only are present in image which means image has low contrast (since only selected range is present). Moreover image is brighter as values are upper range [Upper Value -> White(Bright), Lower Value -> Black(Dark))]

In [ ]:
```
# Calculation of CDF
hist_cdf = [0]*256
print(len(hist_cdf))
for i in range(256):
    if i == 0:
        hist_cdf[i] = histogram[i]
    else:
        hist_cdf[i] = hist_cdf[i-1] + histogram[i]

#Plot CDF
plt.title("Original Image CDF")
plt.bar(idx, hist_cdf, 1)
plt.show()
```

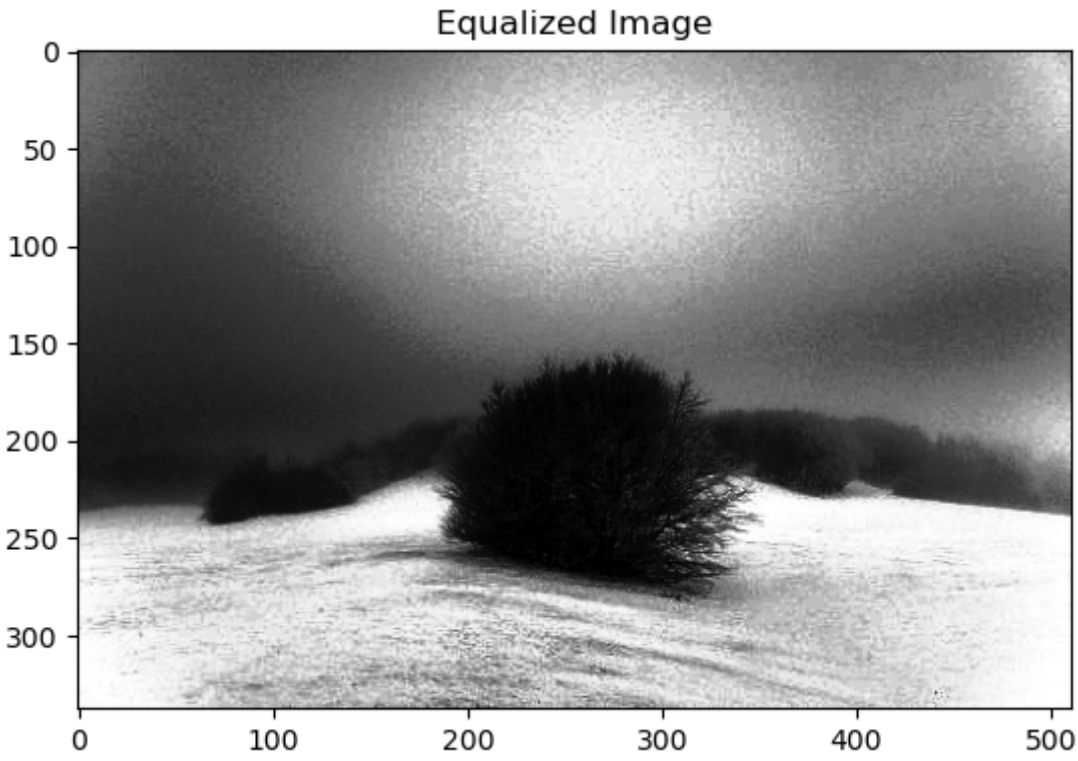
```
256
```



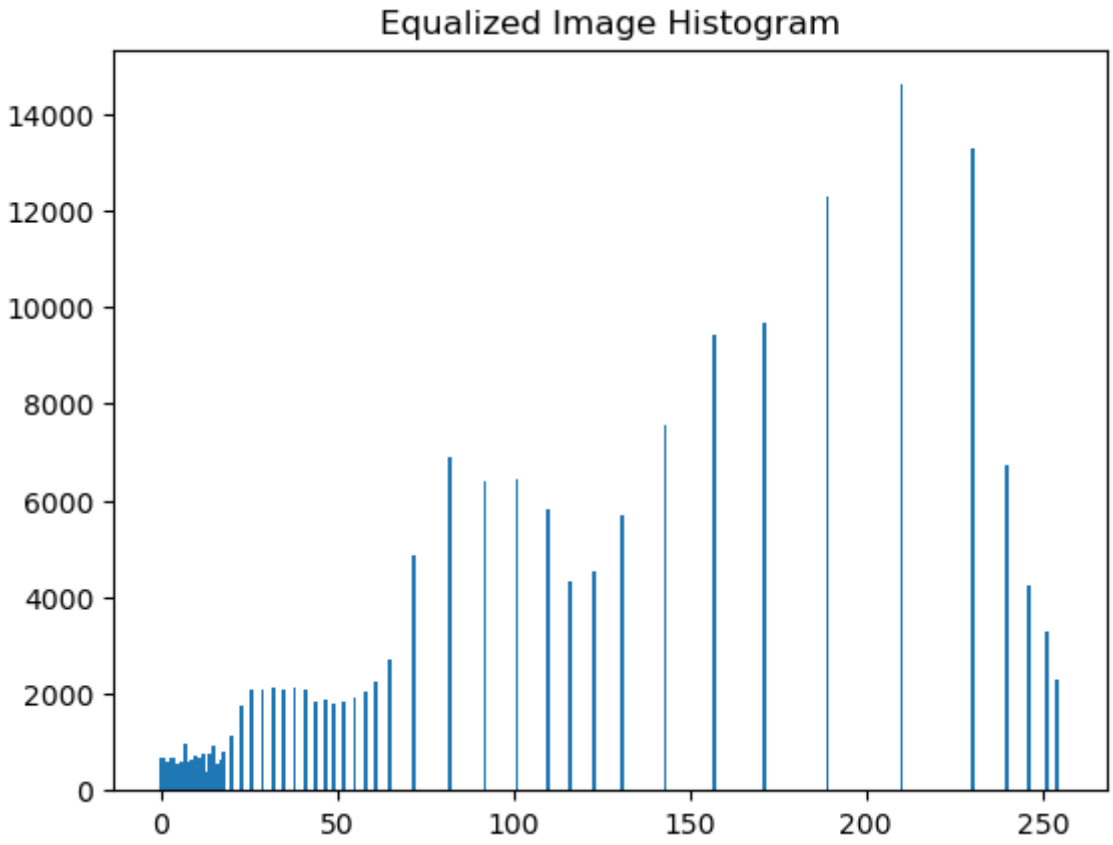
Yes, CDF is appropriate equalization function as in this we want to distribute brightness uniformly over the whole available range instead of only upper range. Hence, since cdf is pixel intensity distribution function and hence we can use CDF as equilization function.

In [ ]:
```
# Apply CDF as Transformation f(x) to image
histogramCDF = np.array(hist_cdf)
hist_cdf_normalized = histogramCDF * 255 / histogramCDF[-1]   #Normalization o
equalizedImage = hist_cdf_normalized[img_gray]   #Applying CDF to image i.e. re

#Display Equalized Image
plt.title("Equalized Image")
plt.imshow(equalizedImage, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fac51455730>
```



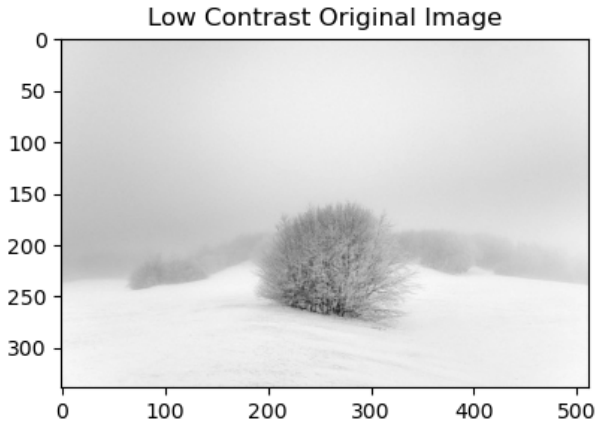
In [ ]:
```
#Equalized Image Histogram
histogram_eq = [0]*256
for rows in equalizedImage:
    for val in rows:
        histogram_eq[int(val)] = histogram_eq[int(val)] + 1

#Plot Histogram
fig, ax = plt.subplots()
ax.bar(idx, histogram_eq, 1)
plt.title("Equalized Image Histogram")
plt.show()
```

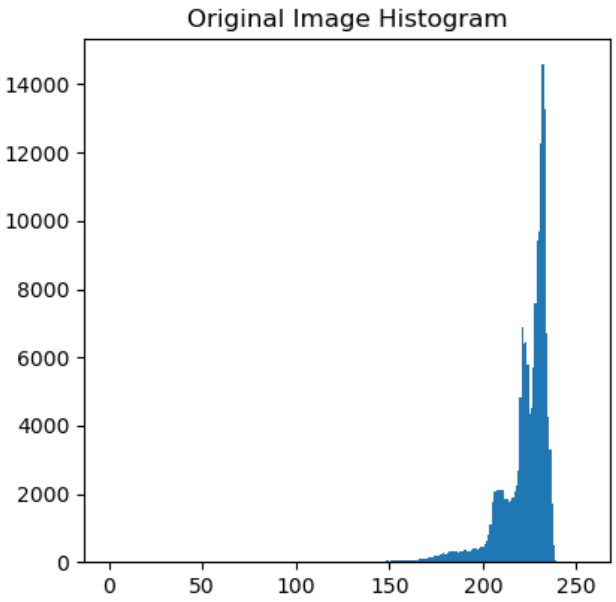


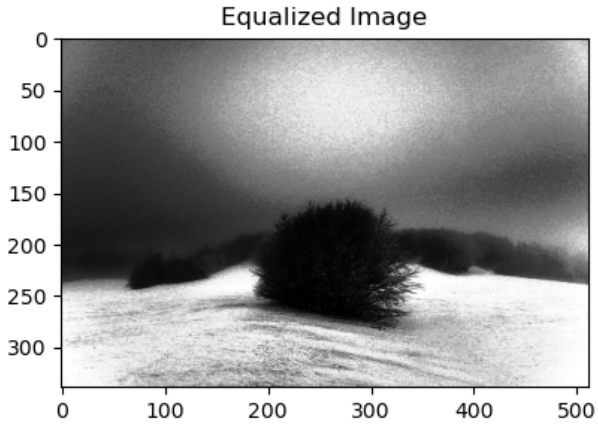
After observing histogram, we can see that values are distributed over the whole range which means image has been equalized.

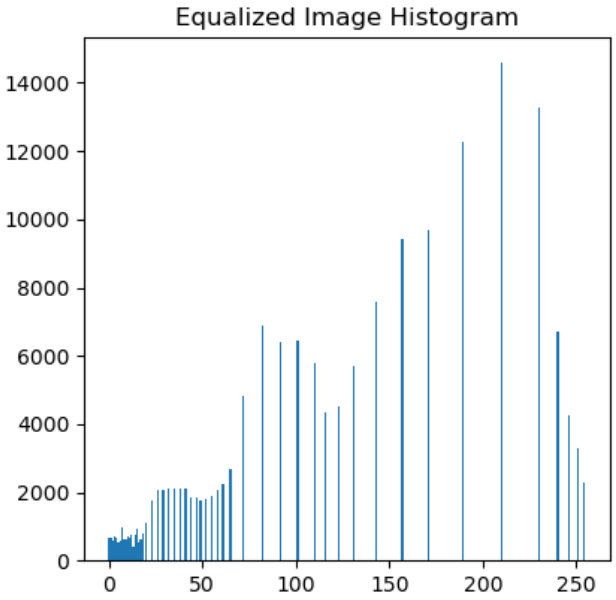
In [ ]:
```
#Results at a Glance

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

plt.subplot(2,2,1)
plt.title('Low Contrast Original Image')
plt.imshow(img_gray, cmap='gray')

plt.subplot(2,2,2)
plt.title("Original Image Histogram")
plt.bar(idx, histogram, 1)

plt.subplot(2,2,3)
plt.title("Equalized Image")
plt.imshow(equalizedImage, cmap='gray')

plt.subplot(2,2,4)
plt.title("Equalized Image Histogram")
plt.bar(idx, histogram_eq, 1)
```

Out[ ]:
```
<BarContainer object of 256 artists>
```









In case of Original image, we can see that image has more white pixels. Moreover, from the histogram we can see that image contains only pixels with higher values.

Whereas in equilized image, we can see that image all kind of pixels from black to white. It can verified by observing its histogram. Values from whole range has been used.