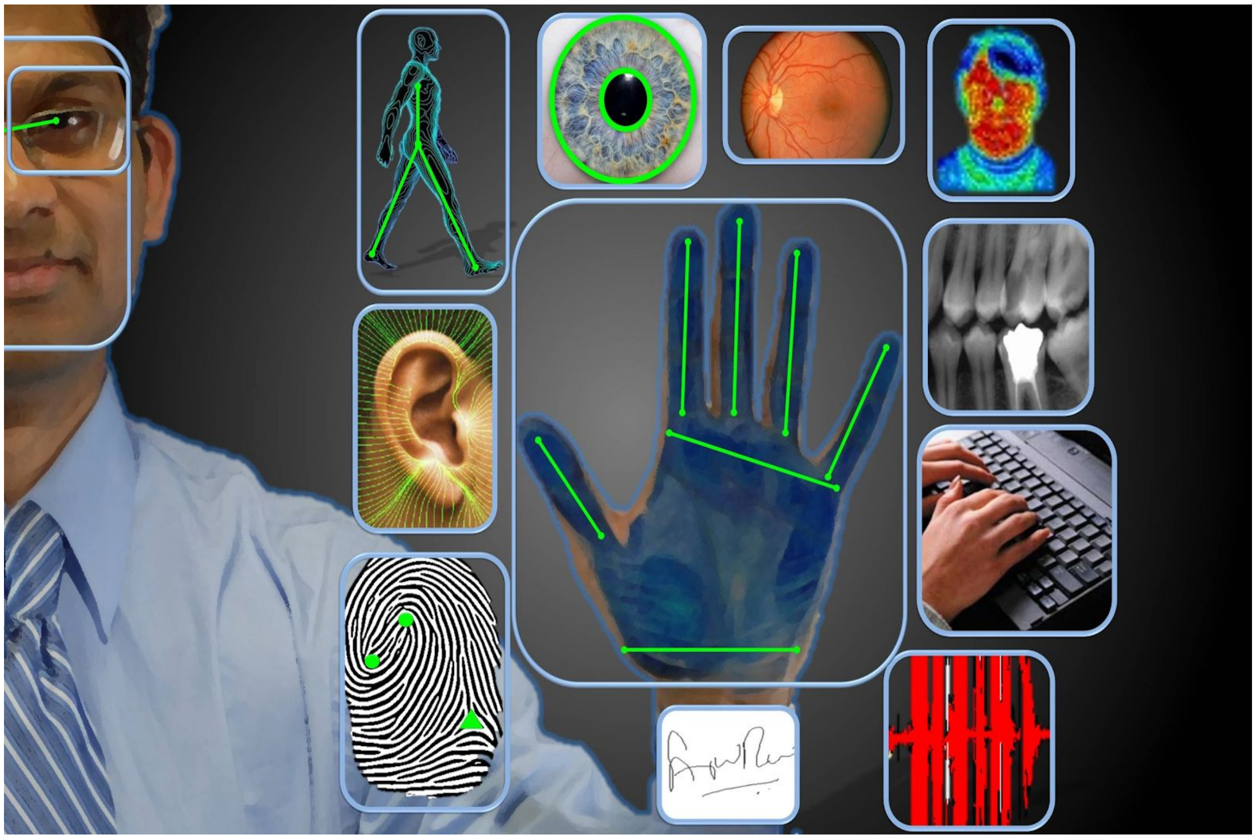


Digital Image Processing

Assignment-1

Atheti Nikhilendra
20161054

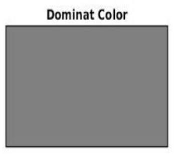
Question-1:

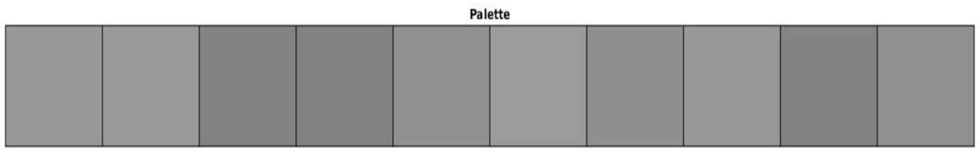
Function: domColors(im,k), displaydomColors(im, domColors(im,k))

Description: domColors takes the image_filename and k{no.of colors } as input and ouputs the list of k dominant colors.displaydomColors() display the output in required manner.

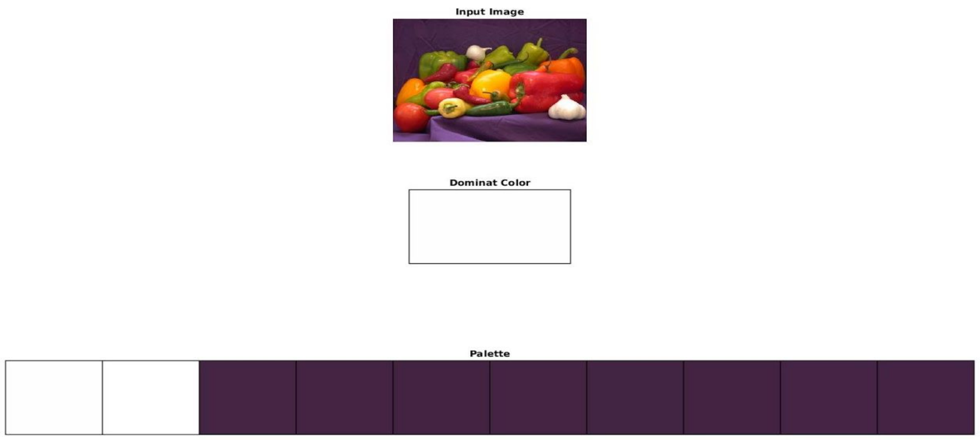
Results on some Images:

‘lena.bmp’ k=10.

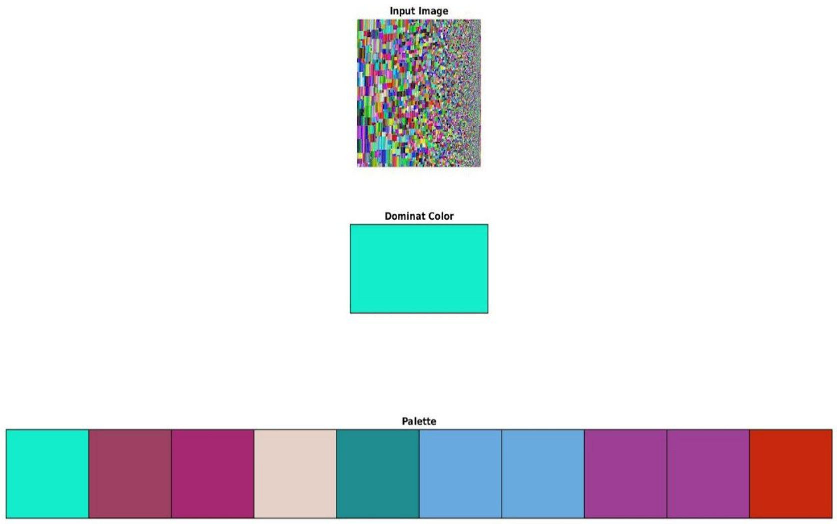




‘pepers.png’ k=10.



‘Q1_allcolors.jpeg’ k=10.



<u>Uses of the above Function:</u>

By knowing the dominant color, we can say that the pixel with dominant color belong to Background of the scene. Used in Classification Techniques.

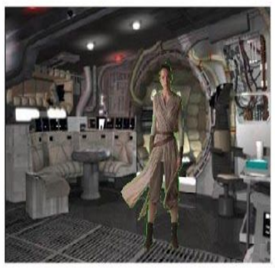
Question-2:

Function: ChromaKey(‘fg’,’bg’,KeyColor)

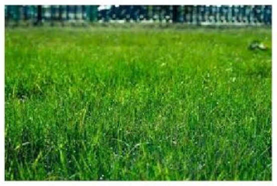
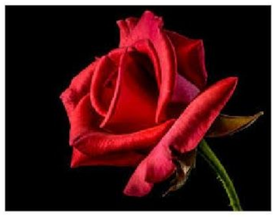
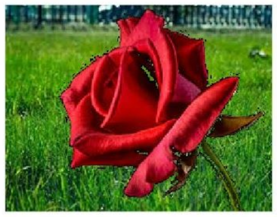
Description: It takes foreground_filename, background_filename, KeyColor as input and shows the output.

Results on Images:

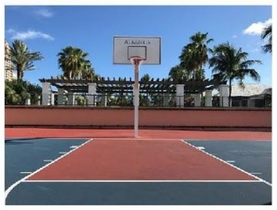
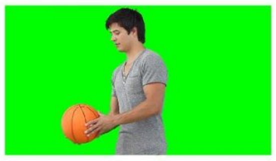
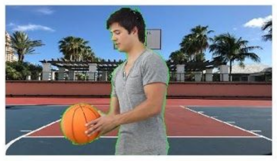
fg.png + bg.png = ‘Q2_givenImage_output.jpg’



rose.jpeg + rose_bg.jpeg = ‘Q2_rose_output.jpg’

q2_input_fg.png + q2_input_bg.png = Q2_output_basketBall.jpg

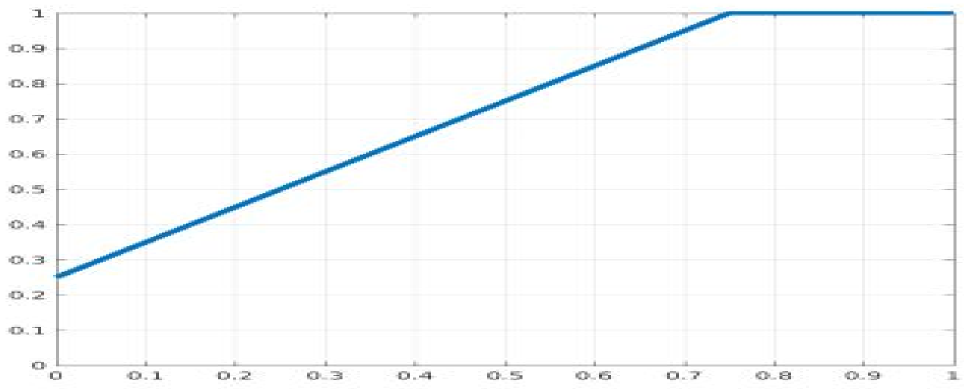
Question-3:

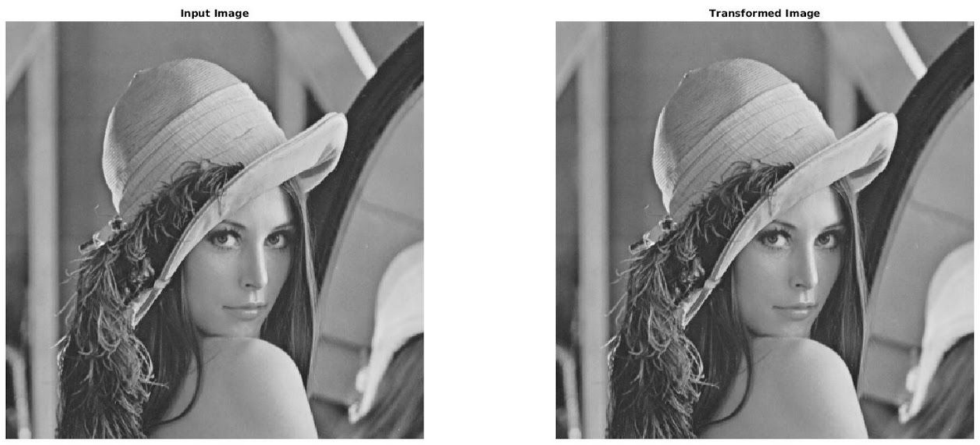
Function: transformIntensity(im, K1,K2,a,b),

Description: It takes the image_filename and the transform Function in the form of 4 vectors K1,K2,a,b. and shows the output of the transform function.

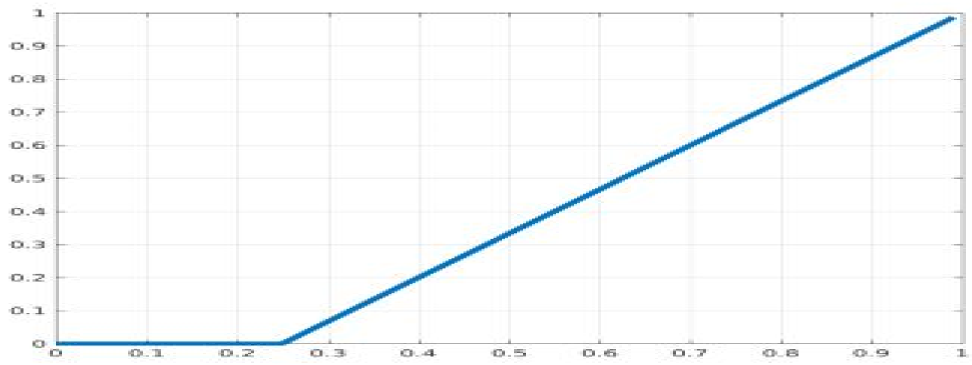
Results for the given Functions:

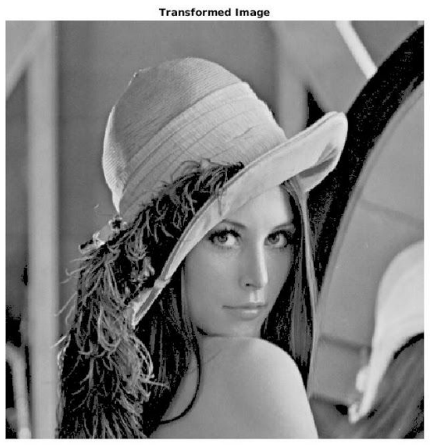
1)K1 = [1,0],   K2 = [0.25, 1],   a = [0,192],   b = [191,255]



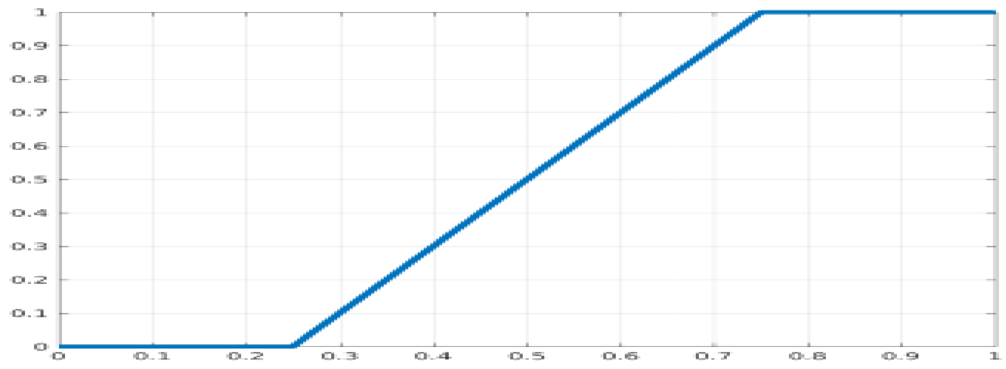


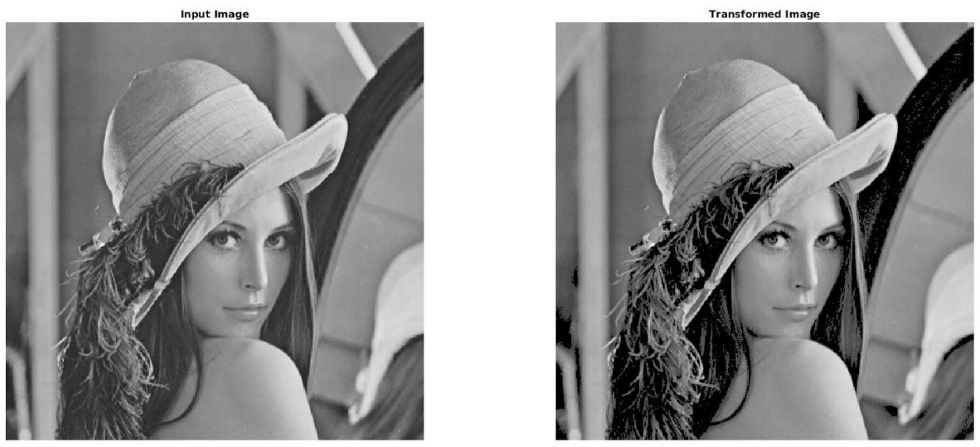
2)K1 = [0,1.33],   K2 = [0, -0.33],   a = [0,64],   b = [63, 255]



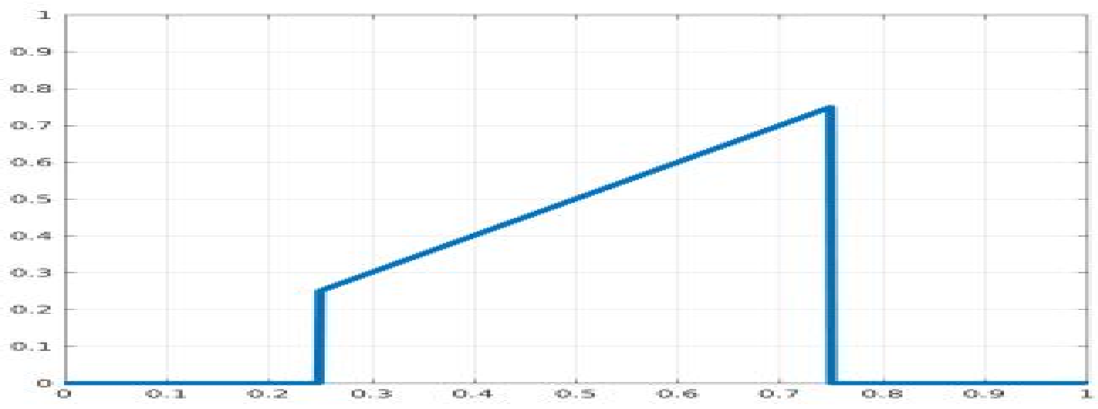


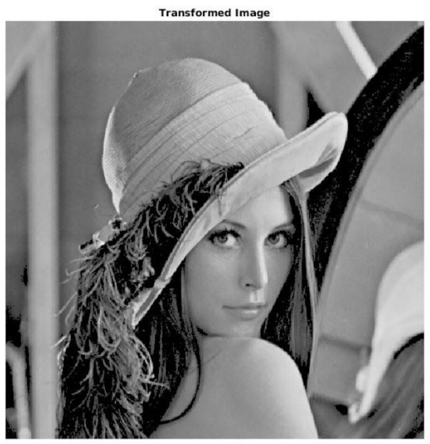
3)K1 = [0,0.5,0],   K2 = [0,-0.125,1], a =[0,64,192]   b = [63, 191, 255]



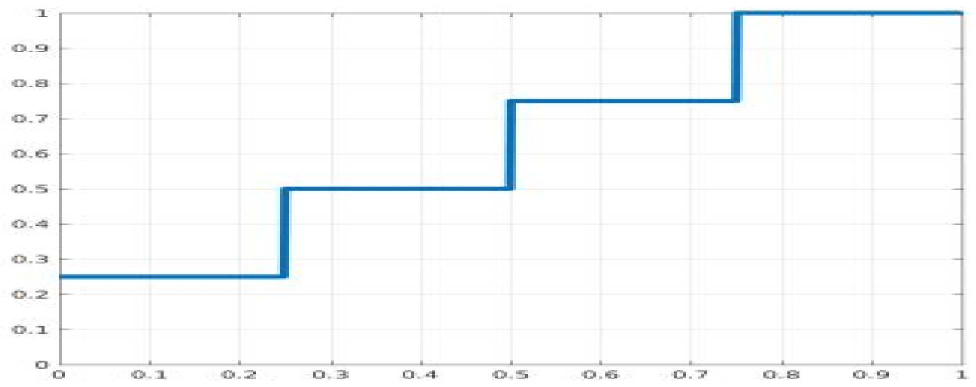


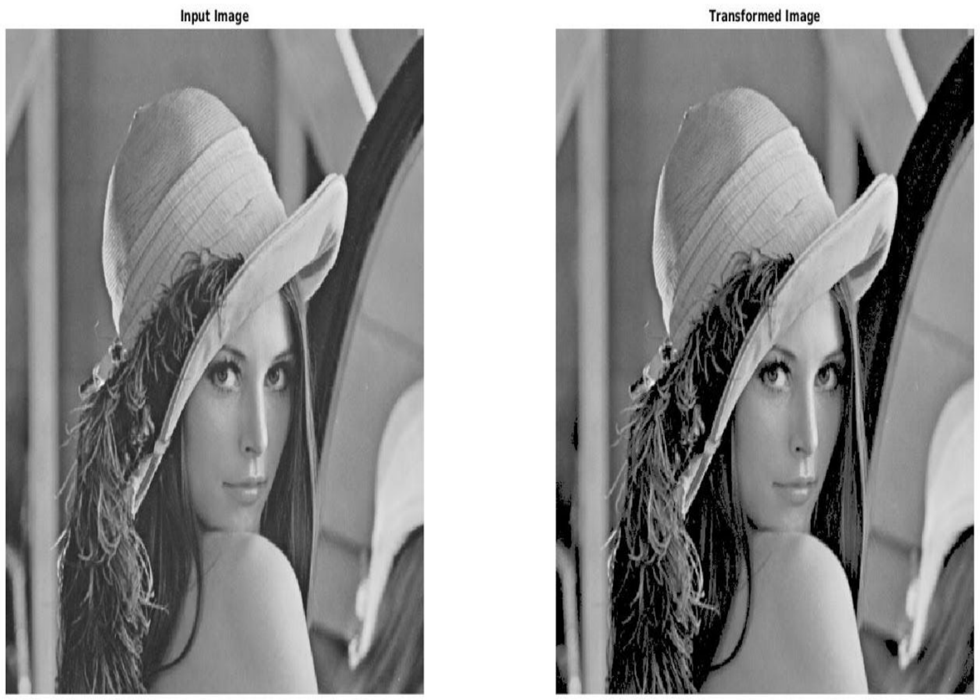
4)K1 = [0,1,0], K2=[0,0,0] a=[0,64,192] b=[63,191,255]





5)K1=[0,0,0,0], K2=[0.25,0.50,0.75,1], a=[0,64,128,192], b=[63,127,191,255]





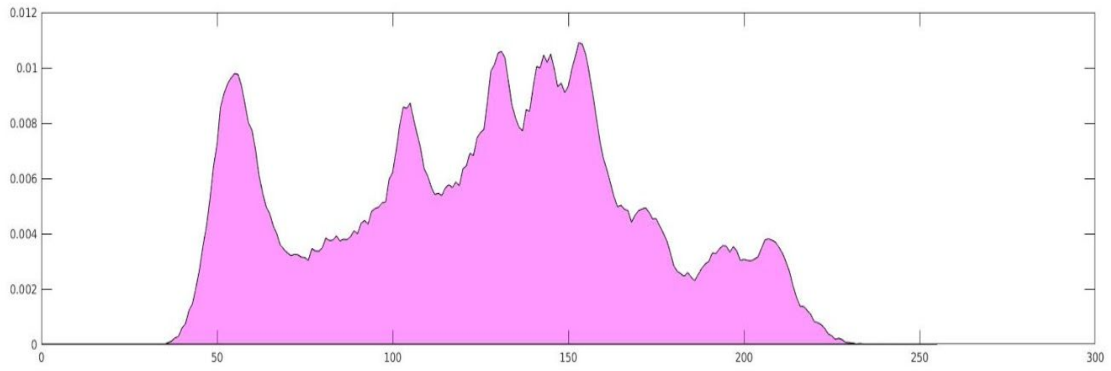
Question-4:

4.1)

Function : myHistogram(im)

Description: It takes the image_filename as input and Plots the Normalized Histograms.

Result on the 'lena.bmp':



4.2)

Here In this Image each pixel can have value in the range[0-255], each value needed **1byte** to store its value because it is ant 'uint8'.

So, NxN image require N*N bytes to store the image.

Whereas in case of Histogram store only 256 values each of **'double'** type so they require **8Bytes** each, On a total Histogram needs only "2048" Bytes of to store the entire Image.

From the Observations in Below Table, we can say that as the size of image changes , The space required by Histogram doesn't change.

Size of image	Bytes to Store Raw Image	Bytes to Store Its Histogram
16x16	16*16	2048
32x32	32*32	2048
64x64	64*64	2048
128x128	128*128	2048
256x256	256*256	2048
512x512	512*512	2048

Question-5:

5.1)

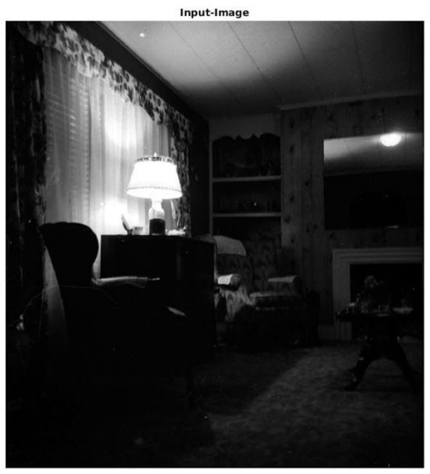
a) **Histogram Equalization:**

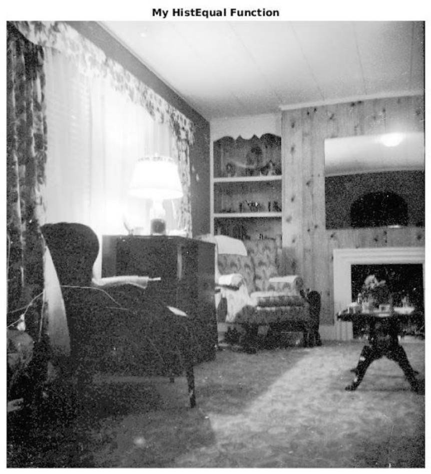
Function : myHistEqual(im)

Description: It takes image_filename as input and the plots the inputs image and Histogram Equalized image in the same figure.

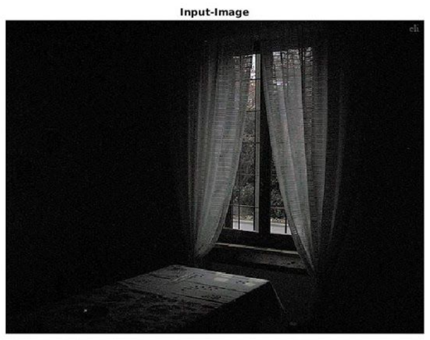
Results on the Given Images:

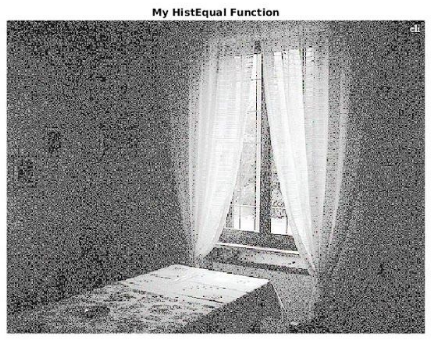
‘hist_equal.jpg’





‘hist_equal2.jpg’





b) **Local Histogram Equalization:**

Function : myLocalHistEqual(im, window)

Description: It takes the image_filename and window size as in input and show the output in the figure.

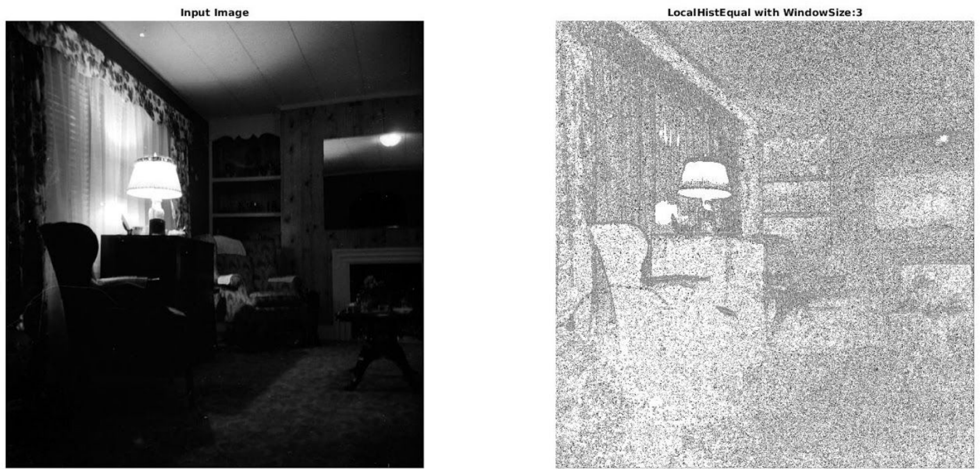
Result:

According to the results as we the increase the window size for the local histogram equalization we get the image more visullay significant.

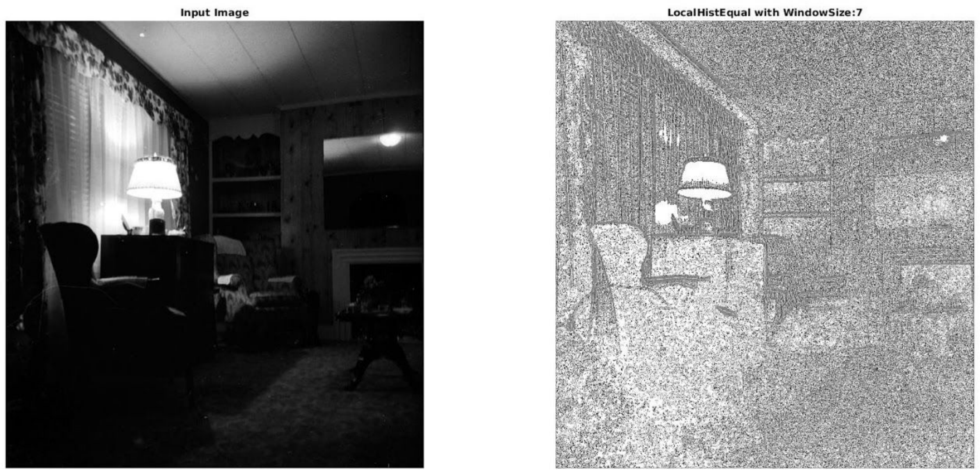
Results on the Given Image:

‘hist_equal.jpg’

window_size=3

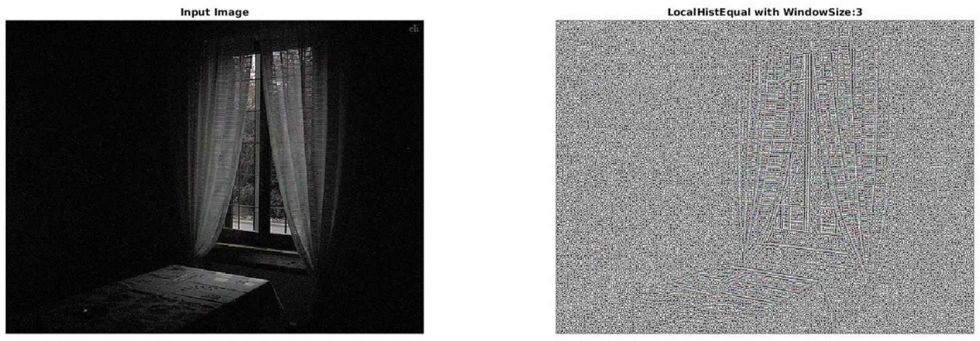


window_size=7

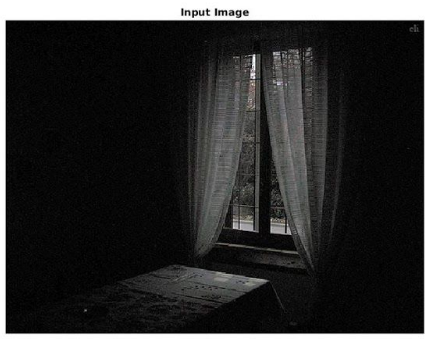


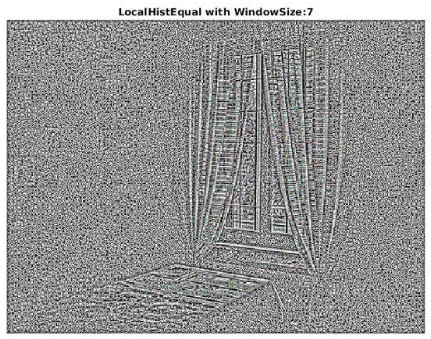
‘hist_equal2.jpg’

window_size=3



window_size=7

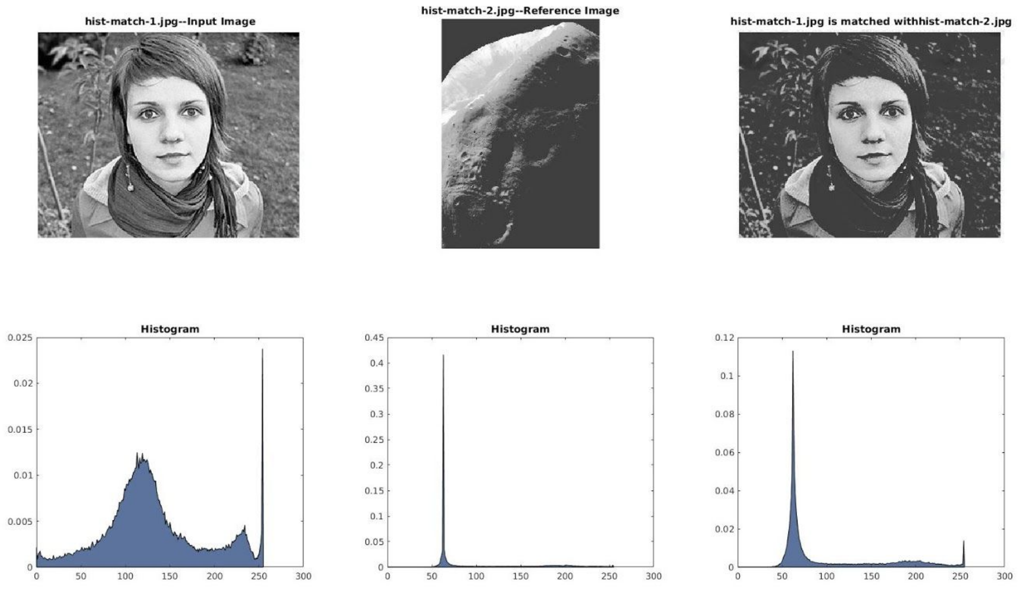




c) **Histogram Matching:**

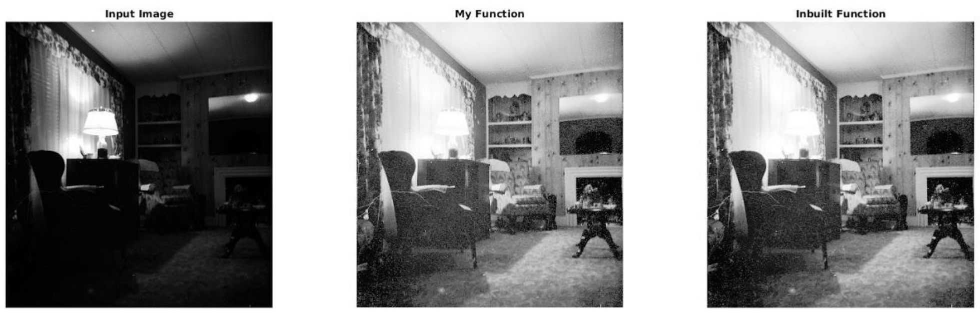
Function: myHistMatching(input_image, reference_image)

Results on Given Images:



5.2)

As compared the both results of my function and the inbulit function, Both are almost same.



Question-6:

6.1)

Functions : BitQuantizeImage(im , k)

Description: It takes the image_filename and k(required Quantization) as input and quantizes the image to the required bits. It outputs the required image{It doesn’t save the image}

6.2)

In general for an n-bit image the valid range of the ‘k’ is [1-n]

6.3)

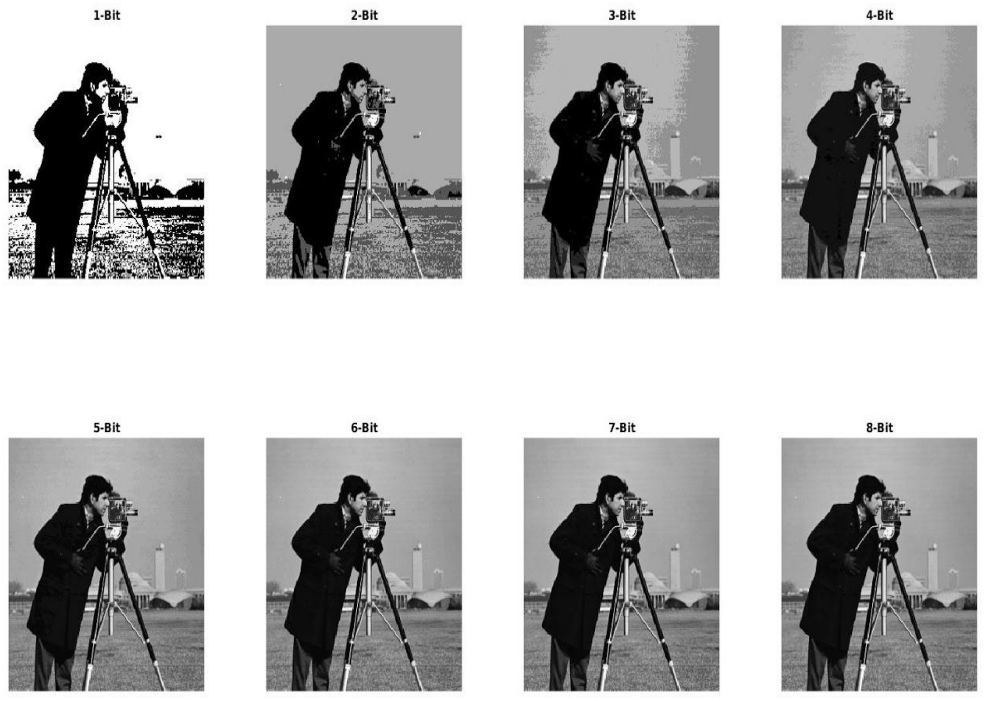
It was included in the function “BitQuantizeImage”

6.4)

Function : displayBitQuantizeImage(im)

Description: It takes the image_filename as input and display the all possible Quantized images. It uses the above implemented function.

Result on the ‘cameraman.tif’:



Question-7:

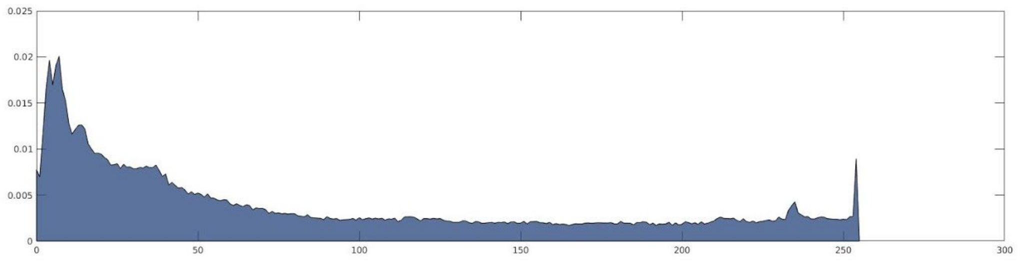
From the below results we can say the following

The intensity values for the Low contrast Image are concentrated in a narrow region thus the difference High and Low is Less. Where as in High contrast Images intensity values are spread over a wide range and the difference between the High and Low intensity is more as compared to Low contrast images

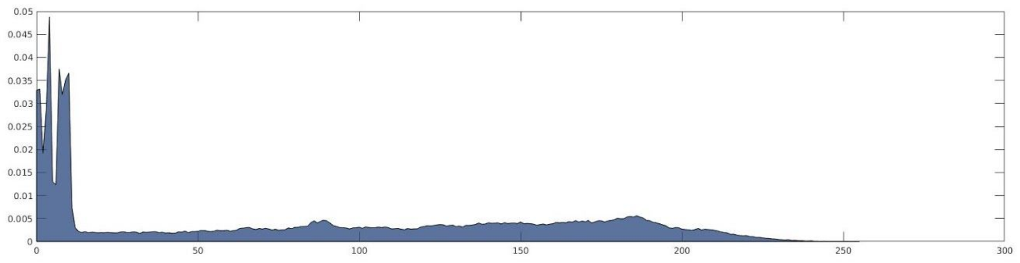
So, High contrast images are visually better than Low contrast images.

Results on differnt Images:

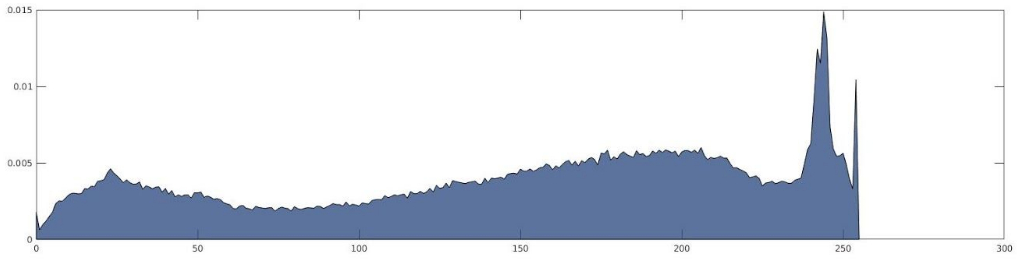
‘Q7_High1.jpg’:



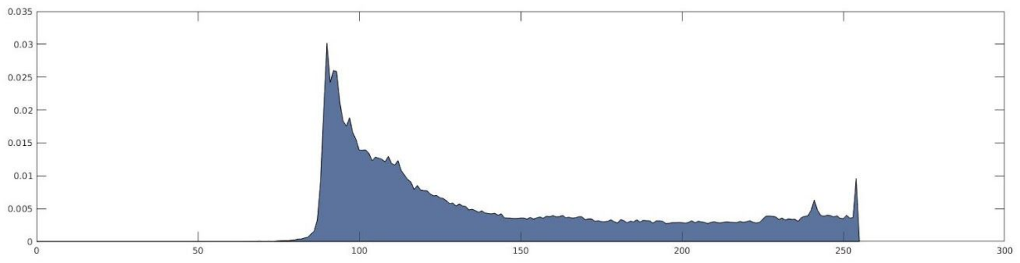
‘Q7_High2.jpg’:



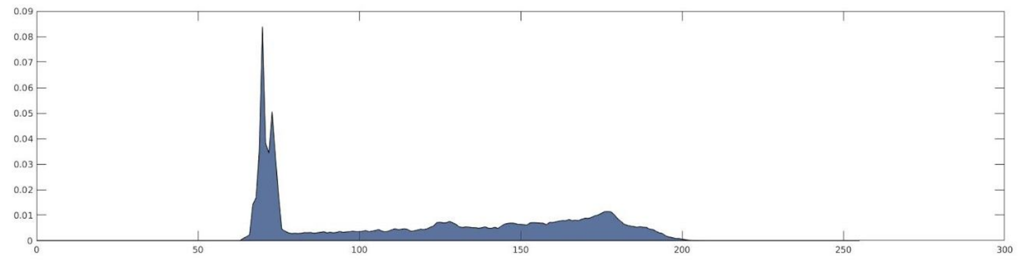
‘Q7_High3.jpg’:



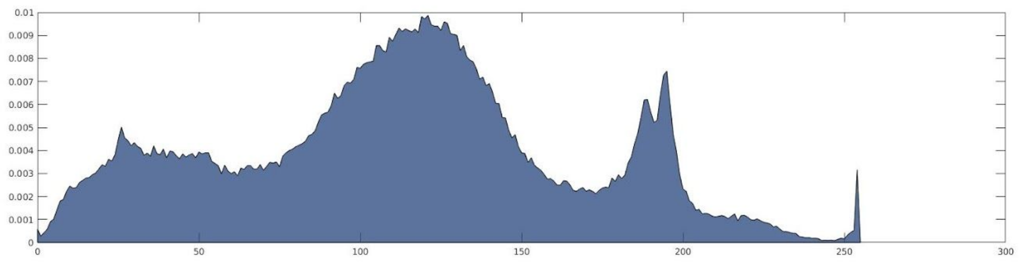
‘Q7_Low1.jpg’:



‘Q7_Low2.jpg’:



‘Q7_Low3.jpg’:



Question-8:

Function name: gsto8bitimage(im)

Description:

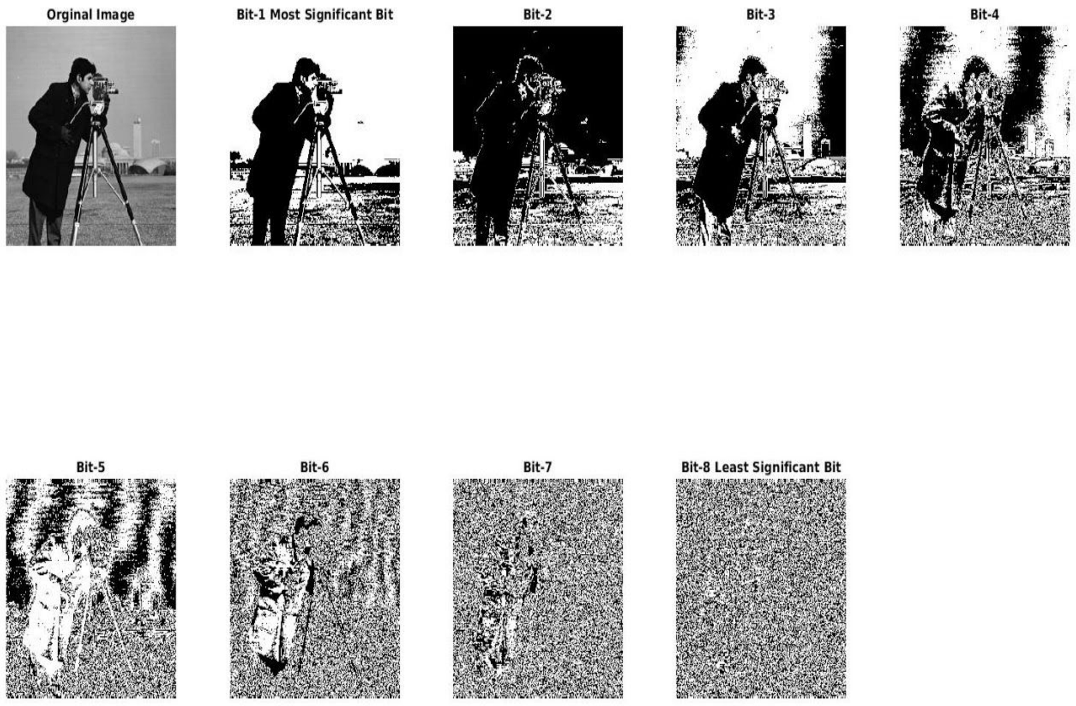
It takes the 8-bit gray scale image as input, Bit Slices the image and shows the corressponding 8 planes in different subplots of a figure.

Result:

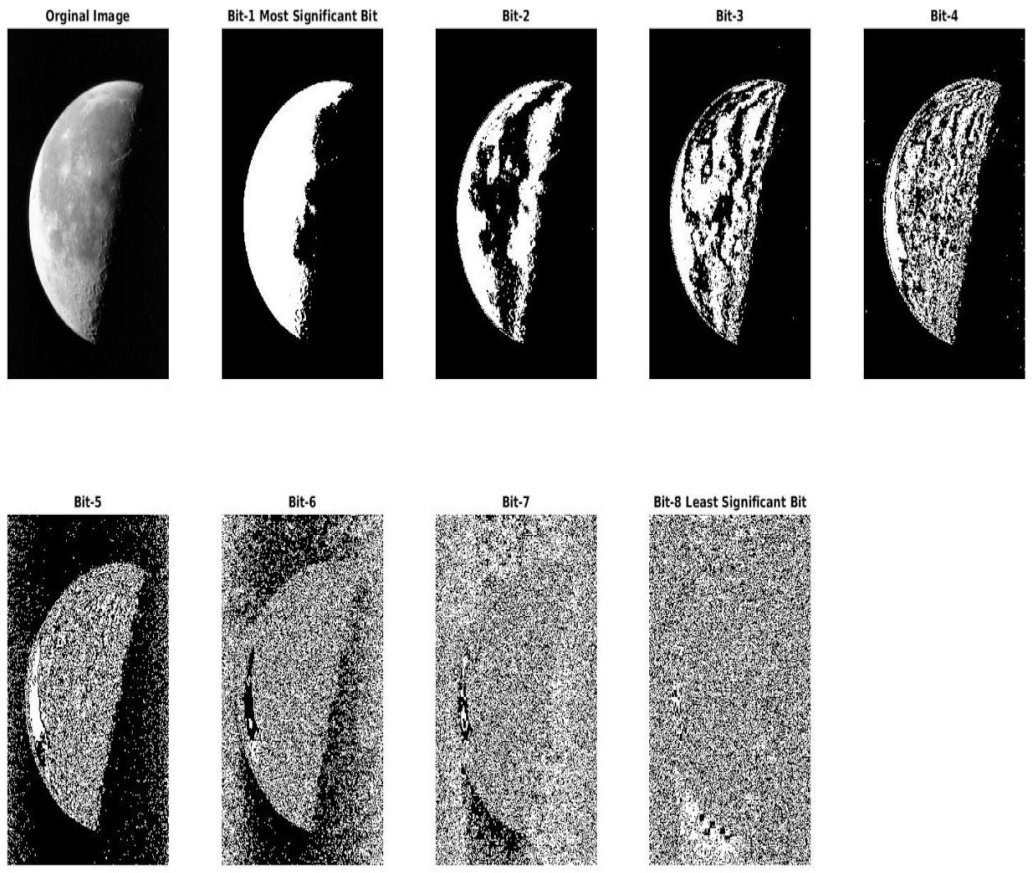
According to the generated results we can say that the Most Significant Bit Planes contain the majority of the important information{ especially the most significant 4 bits}. The remaining bit planes contribute to the less relevant information.

Results on the some images:

Input_image: cameraman.tif



Input_image: moon.tif



Question-9:

Function: BinarizeImage(im, [‘GOT’ | ‘LAT’ ,{0-1}]),

displayBinarizeImages(im)

Description: Takes the Input Image and Thresholding type {‘GOT’, ‘LAT’},

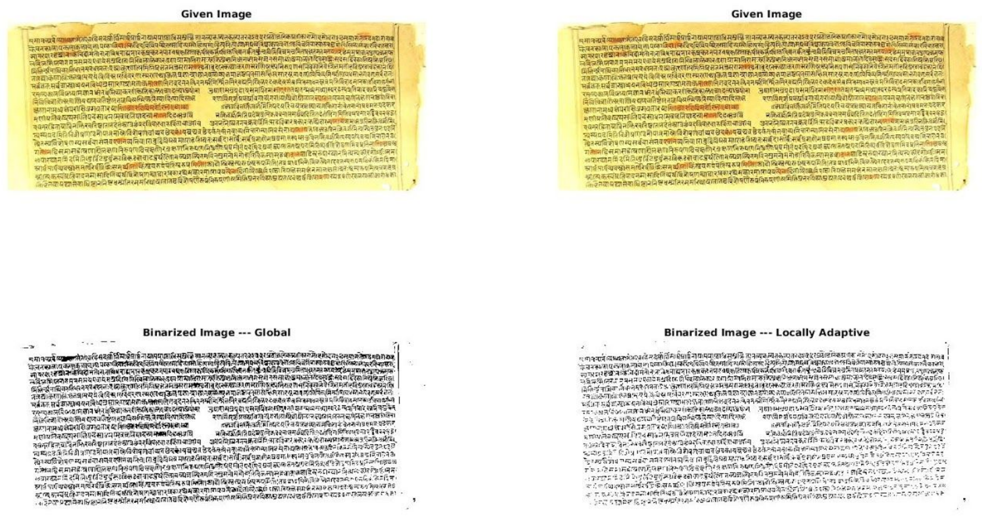
Display function show the both output of both Binarized images.

Results on Given Images:

palm-leaf-1.jpg

Given Image
Given Image
Binarized Image --- Global
Binarized Image --- Locally Adaptive

palm-leaf-2.jpg


Given Image
Given Image
Binarized Image --- Global
Binarized Image --- Locally Adaptive

Global Thresholding: Choosing a Threshold that seperates the background and the Object in the image based on the Histograms.

Locally Adaptive Thresholding: Divide the Image in to Local Regions and apply global Thresholding in each Local Region based on Histograms

Locally Adaptive Thresholding gives better performance incase of Dark Images or Images having Shadows, then Global thresholding

Global Thresholding gives better performance in Bright Images and Where there is a more difference between background and Object Pixels.