DIP Assignment 1

Sagrika Nagar
20171204
B. Tech ECE (3rd Year)

Q 1

Part I (filename: most_frequent_color.m)

The name of the function is most_frequent_color. It takes the image as input (im) and returns the most frequently occurring color in that image (most_frequent).

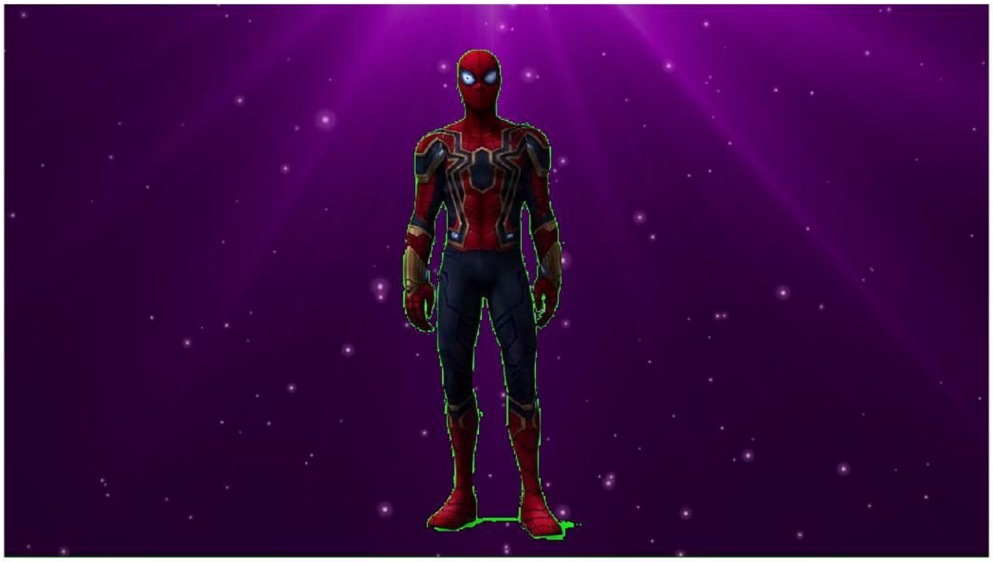
Eg: The most frequent color in fg.jpg is [21, 255, 8].

Part II (filename: q1.m)

The name of the function is mergeImage. It takes the foreground image (fg) and background image (bg) as its inputs.

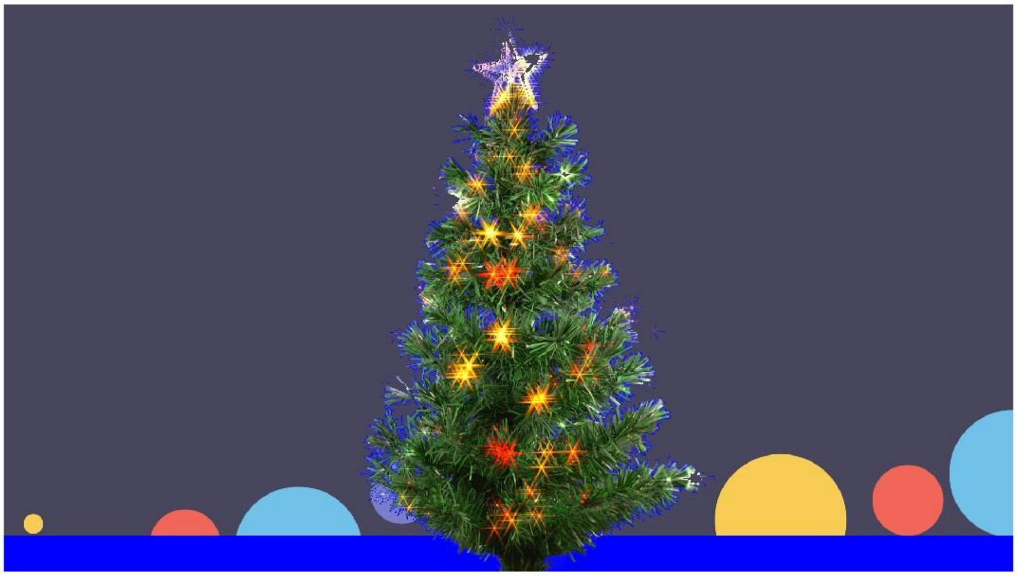
Inputs: *fg.jpg* and *bg.jpg* were passed as fg and bg respectively.

Output Image (*merged1.jpg*):

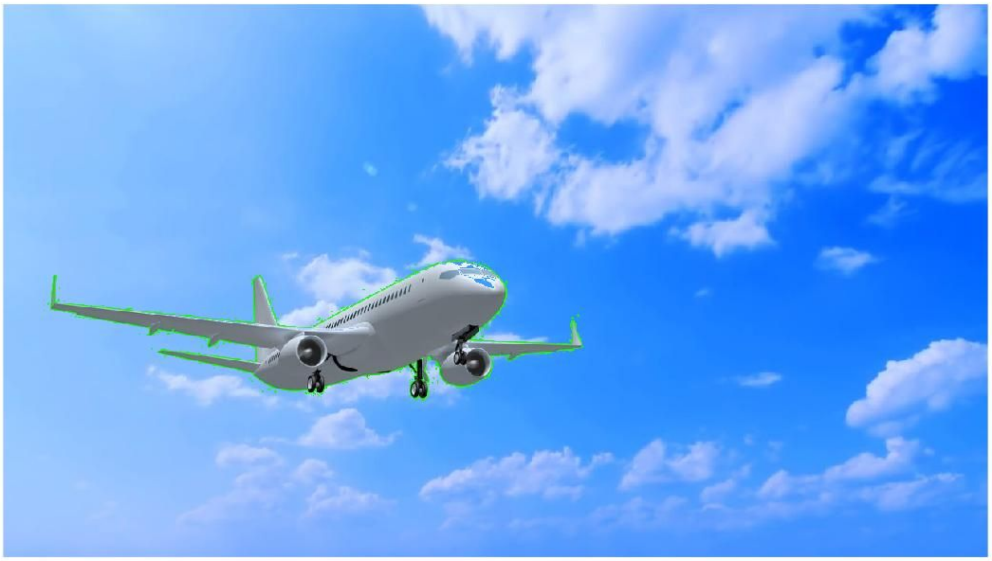


Part III (q1.m)

Inputs: *fg2.png* and *bg2.png*
Output Image (*merged2.jpg*):



Inputs: *fg3.jpg* and *bg3.jpg*
Output Image (*merged3.jpg*):



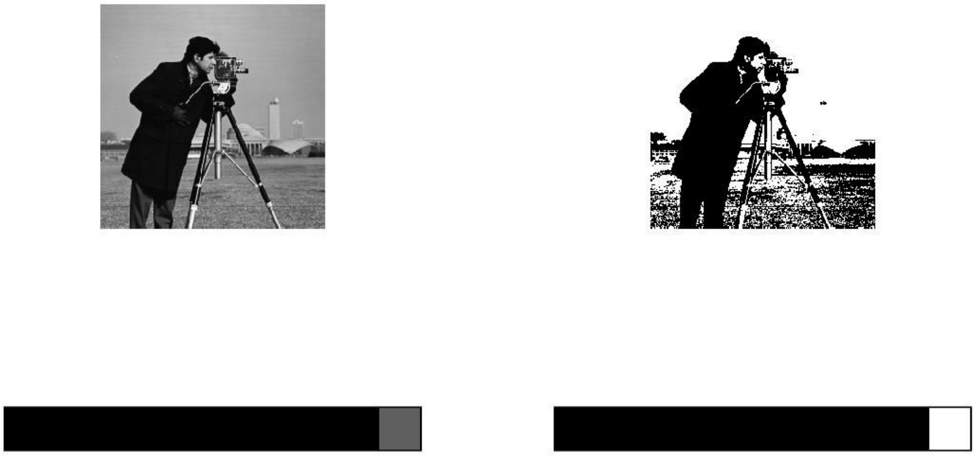
Q2

Part I (linContrastStretching.m)

The function linContrastStretching takes a grayscale image im and a and b as inputs. The resulting image (final) has intensity range as [a,b].

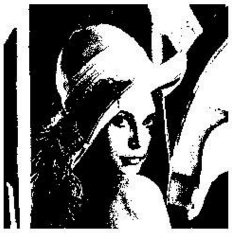
Part II (displayColorbar.m and q2.m)

The function displayColorbar takes the image and k as inputs. k is the number of most frequently occurring colors wanted in the colorbar. Input: '*cameraman.png'*, Output: '*cameraman_colorbar.jpg'*



Part III (q2.m)

Input: *lena.jpg,* Output: *lena_colorbar.jpg*



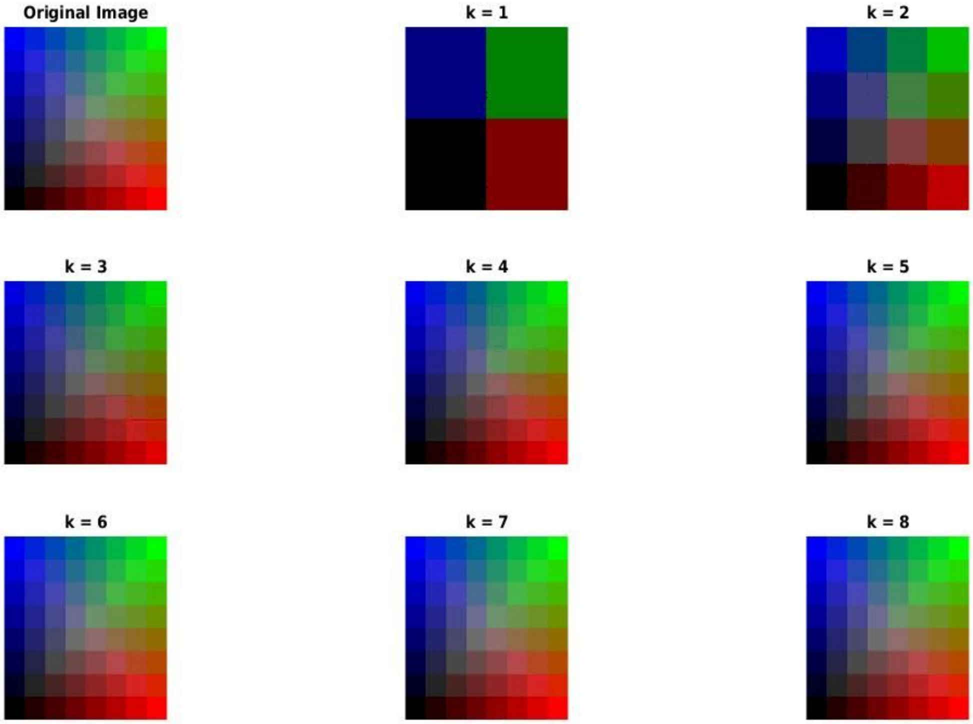
The effect is more on images with lesser contrast i.e. small range of intensity values. It's lesser on images whose intensity is spread out. This is because contrast stretching increases the range of intensity values. So the change in intensity values of images with lesser contrast is much higher than ones with more contrast.

Q3

Part I (BitQuantizeImage.m, q3a.m)

The function BitQuantizeImage takes an 8-bit image (im) and the number of bits the image is to be quantized to (k), as inputs. It returns the k-bit quantized image (quant_im).

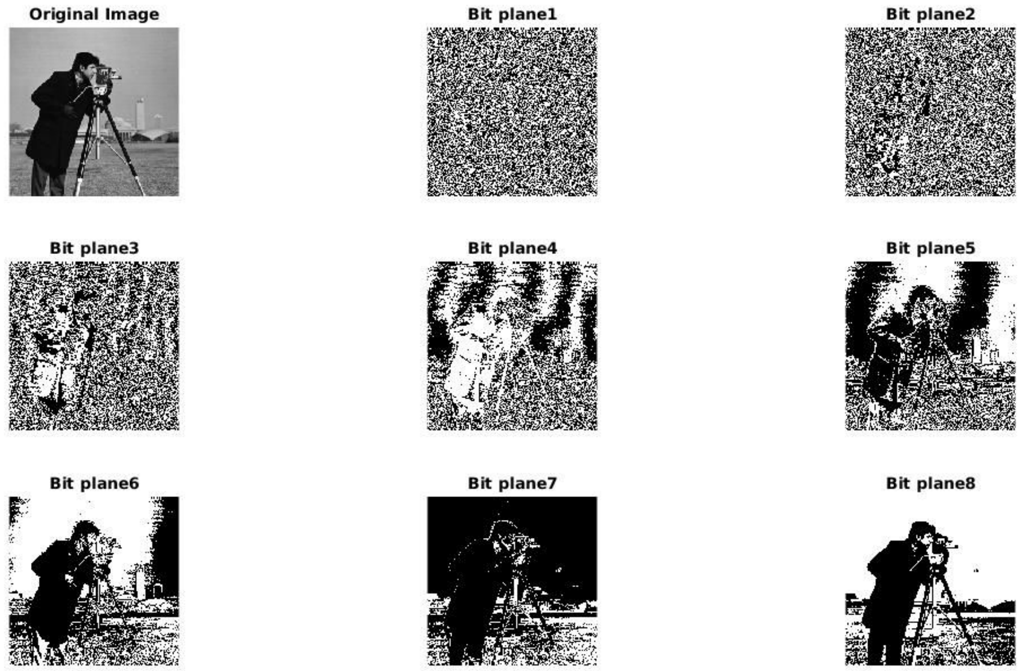
Input: *quantize.jpg* , Output: (*quantize_output.jpg*)



Part II (q3b.m)

The code displays the different bit planes of an 8-bit image *‘cameraman.png’*.

Output: *(bitplanes.jpg)*



Part III

Given *lena.jpg*, Operations applied on:

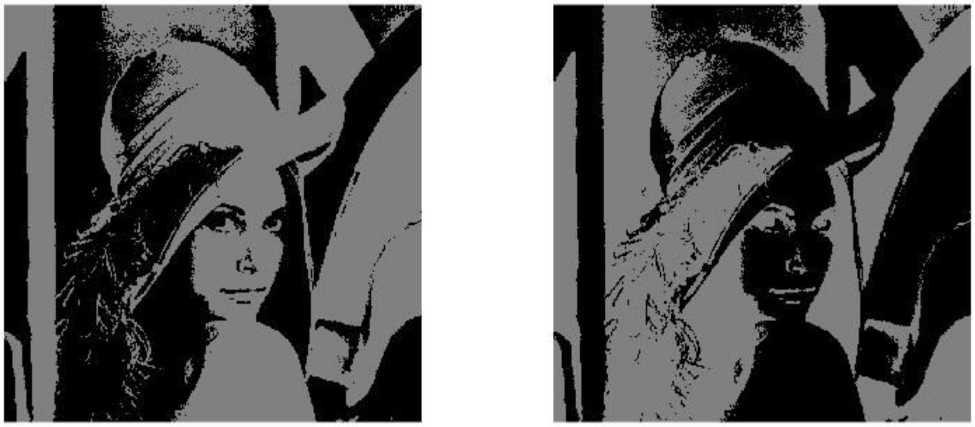
1. *lena1.jpg:* Bit plane Slicing. Bit plane 5
2. *lena2.jpg:* Bit Quantization to 2 bits
3. *lena3.jpg:* Bit plane slicing: Bit plane 8

Q 4

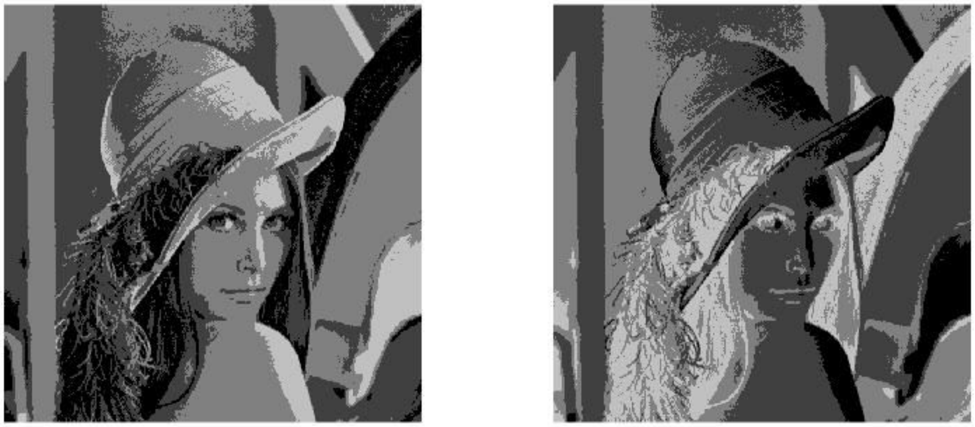
Part I (NegativeTransformation.m, q4a.m)

The function NegativeTransformation takes image(im) and the maximum intensity(max_intensity) of the image as inputs. It returns negatively transformed image(neg). Output for 8 k-bit quantized images of *lena.jpg*:

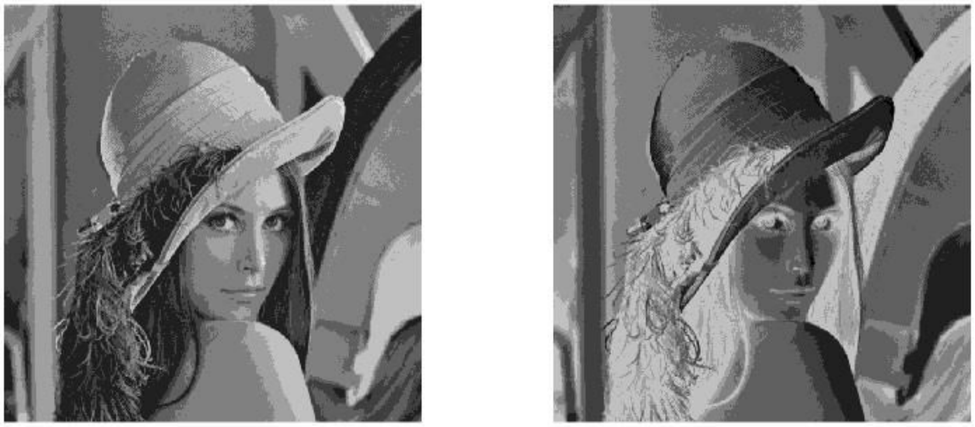
k = 1, *lena_1.jpg*



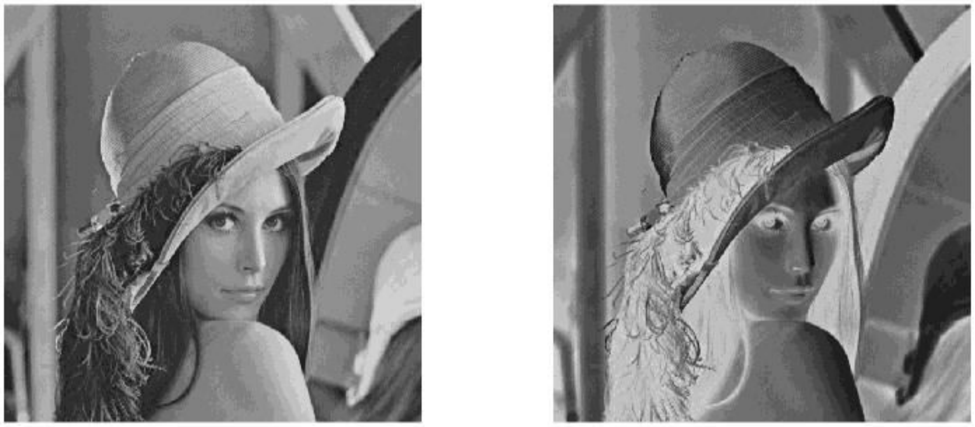
k = 2, *lena_2.jpg*



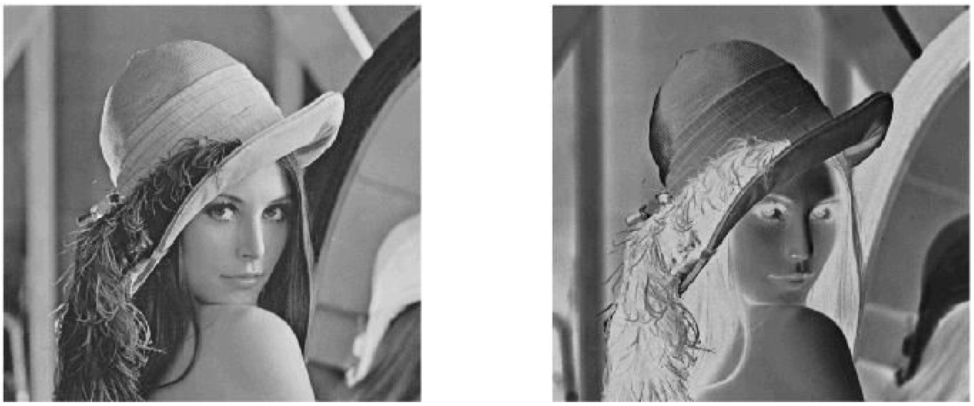
k=3, *lena_3.jpg*



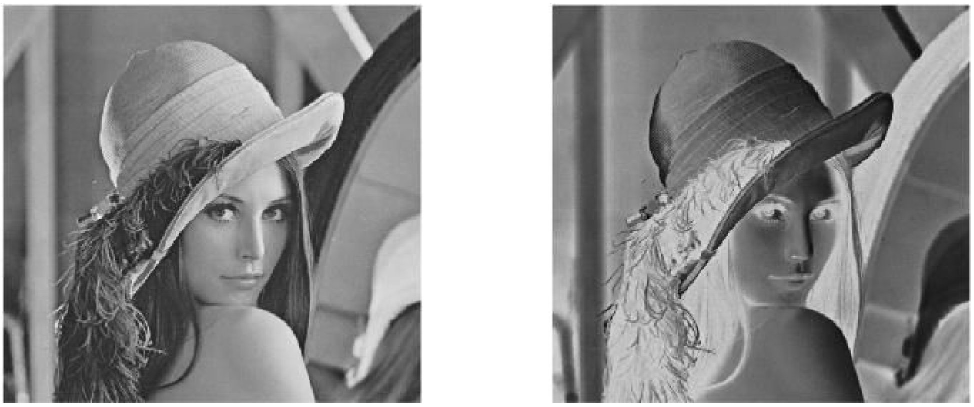
k=4, *lena_4.jpg*



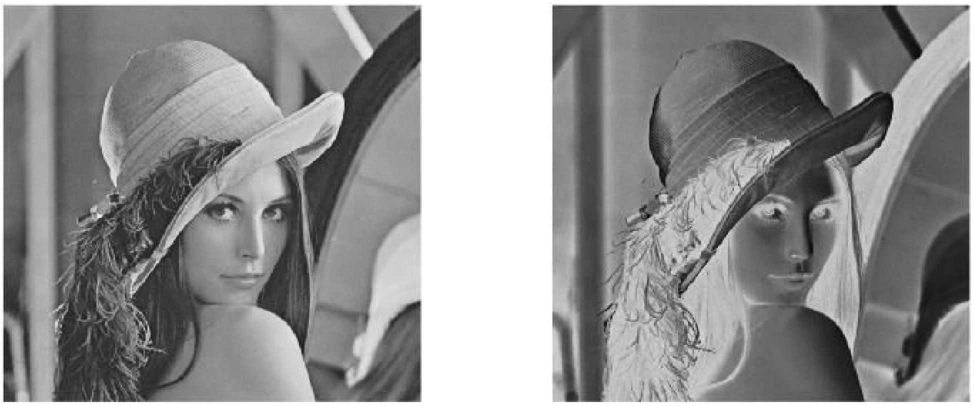
k=5, *lena_5.jpg*



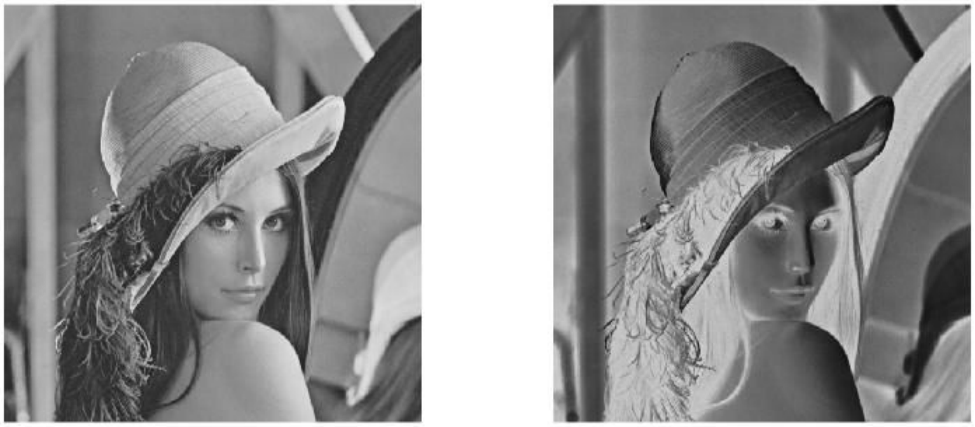
k=6, *lena_6.jpg*



k=7, *lena_7.jpg*

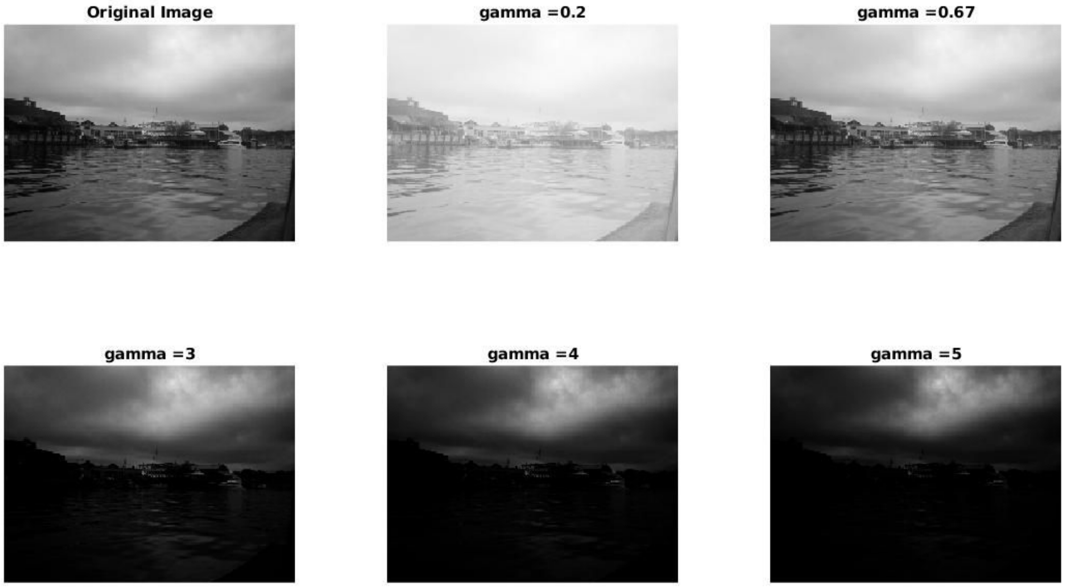


k=8, *lena_8.jpg*



Part II (gamma_corr.m)

The code applies gamma transformation on image *‘gamma-corr.png’* with γ = 0.2, 0.67, 3.0, 4.0, 5.0. Results (*gamma-corr_output.jpg*):



Original Image
gamma =0.2
gamma =0.67
gamma =3
gamma =4
gamma =5

It is observed that when gamma < 1, the transformed image is brighter and when gamma>1, the transformed image is darker.

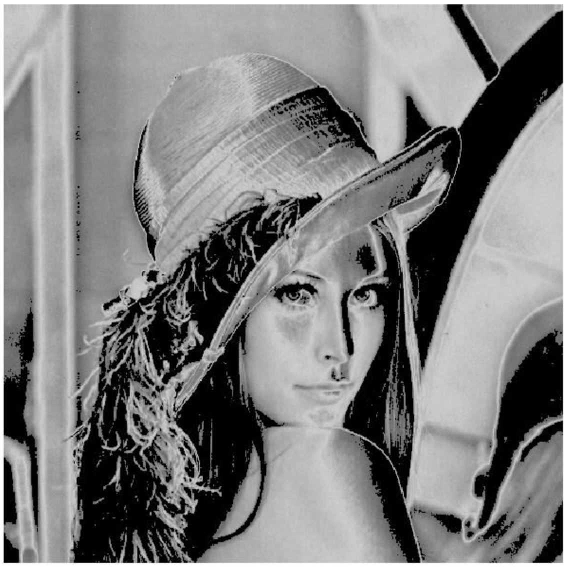
Part III (PiecewiseTransform.m, q4c.m)

The function PiecewiseTransform takes image(im) and k1, k2, a, b vectors as inputs such that:

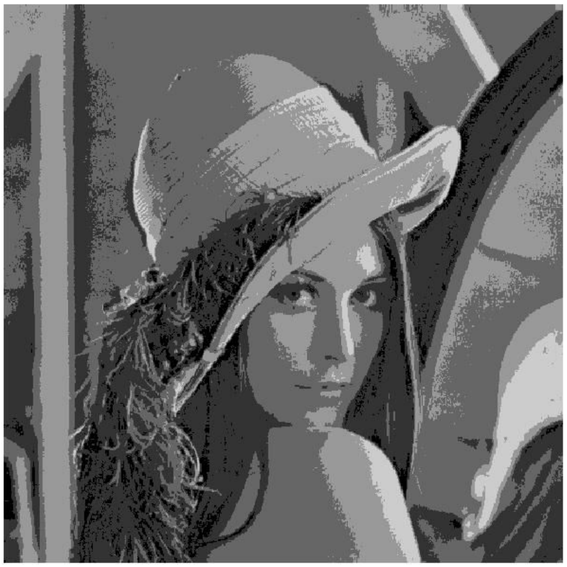
Output = k1*im + k2

for *lena.jpg*, the piecewise transforms are:

(i) k1 = [0, 4/3, -2, 0]; (ii) k1 = [0, 0, 0, 0, 0];
k2 = [0, 0, 2, 0]; k2 = [0, 0.2, 0.4, 0.6, 0.8];
a = [0, 0.3, 0.6, 0.8]; a = [0, 0.2, 0.4, 0.6, 0.8];
b = [0.3, 0.6, 0.8, 1]; b = [0.2, 0.4, 0.6, 0.8, 1];



piecewise1.jpg



piecewise2.jpg

Q 5

1) If the MSB bits in the bitplane are set to zero, the histogram changes. The histogram is cut into two halves, second half of the histogram is stacked on top of the first half. The histogram values after the mid point are zero.

2) If the LSB bits are set to zero, the histogram is almost similar to the original one. It becomes less dense.

3) Since the intensity of the grayscale image ranges from [0, 255], the pixels are defined by 8 bits. Therefore, each pixel needs 10 bits to be transmitted.
   No. of pixels = 512 x 512 = 262144
   Time required over 56K baud link = $\frac{262144 \ x \ 10}{56000}$ = 46.81 sec

   Time required the same image over 3000K baud link = $\frac{262144 \ x \ 10}{3000000}$

= 0.87 sec

Q 6

Part I

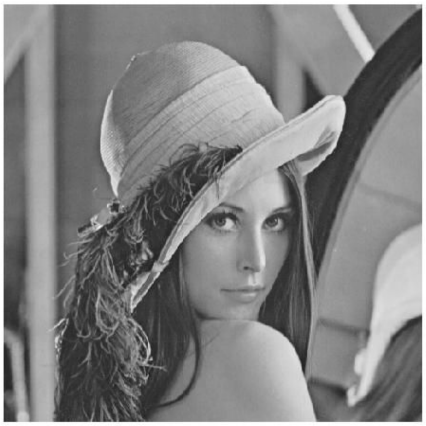
histEqualization.m

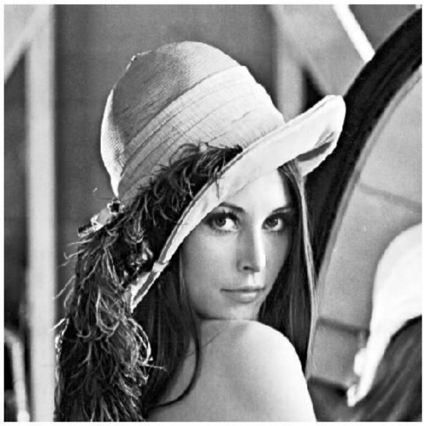
The function histEqualization takes a grayscale image im and applies histogram equalization on it.

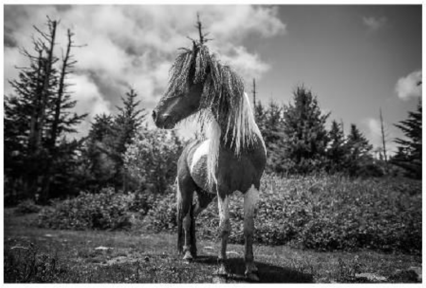
Part II (q6b.m)

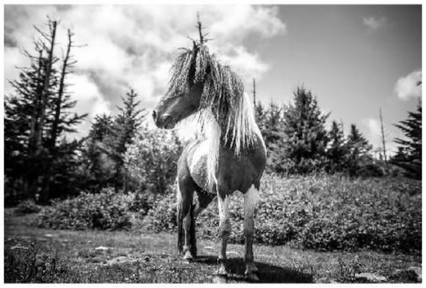
Inputs : *lena.jpg, horse.jpg*

The output is saved as *lena_hist.jpg and horse_hist.jpg*.





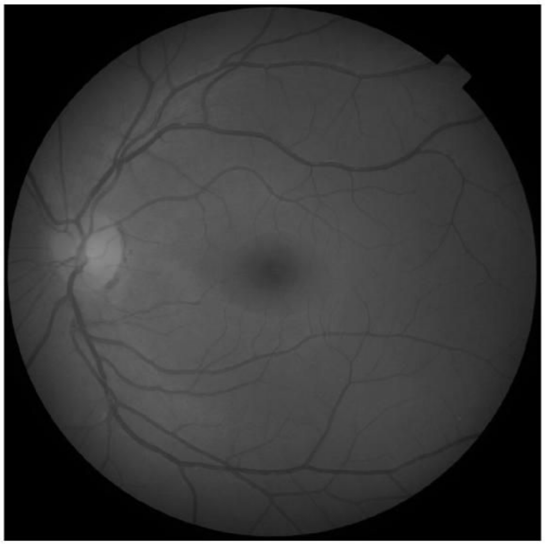




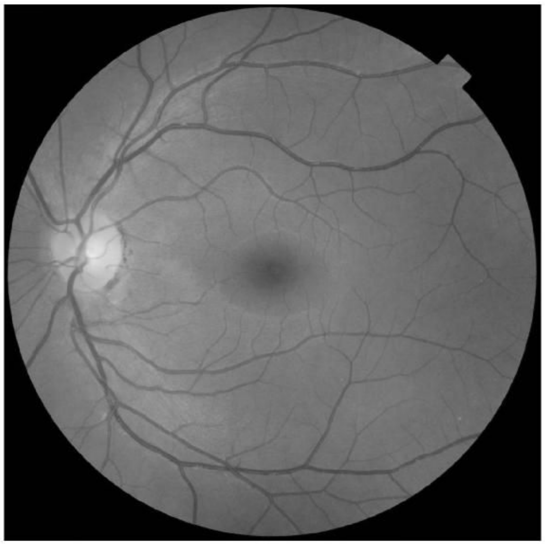
The transformation spreads out the intensity values throughout the histogram of the image and hence giving more contrast to the image.

Part III (histMatching.m, q6c.m)

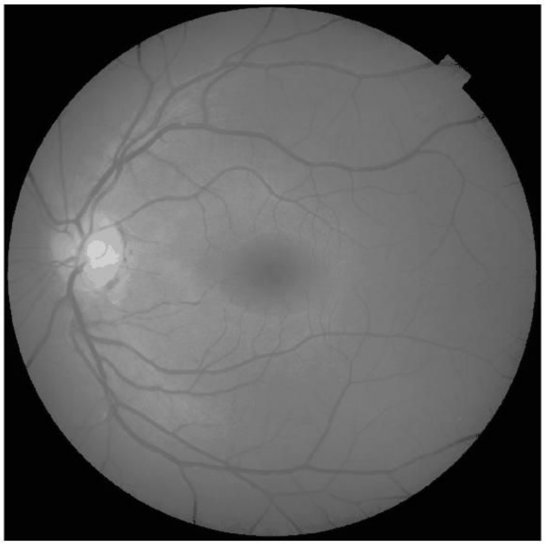
The function takes an image (im) and a reference image (ref) and matches the histogram of im with that of ref.



im.jpg



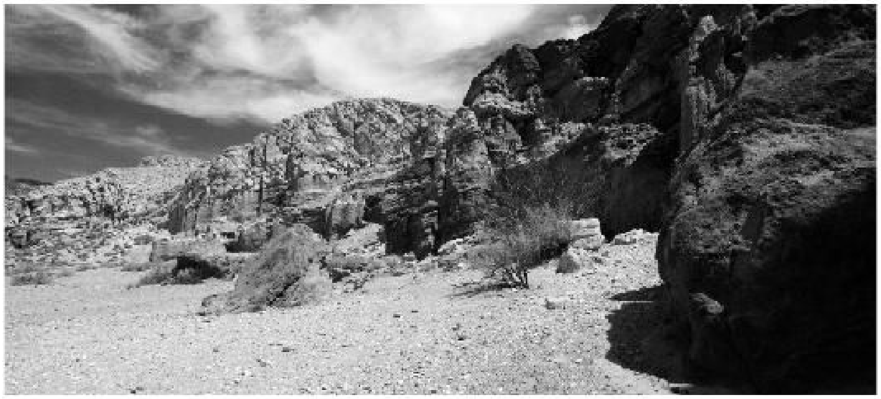
ref.jpg

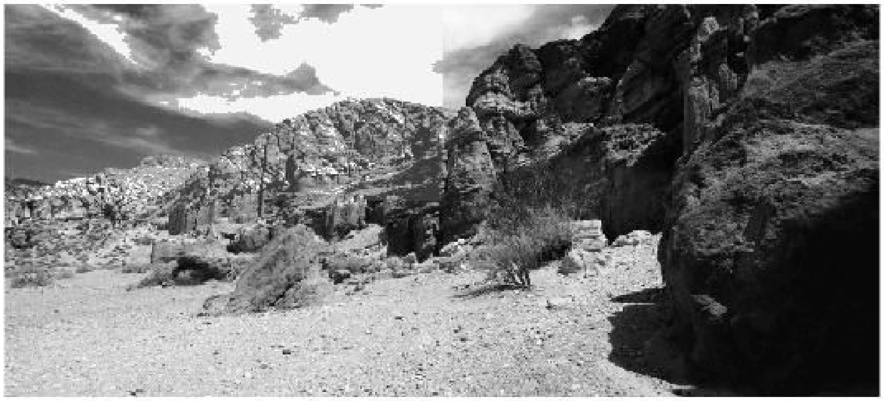


histmatch.jpg

Part IV (Q6d.m)

The input image (*canyon.jpg)* and the output image(*canyon2.jpg) are:*

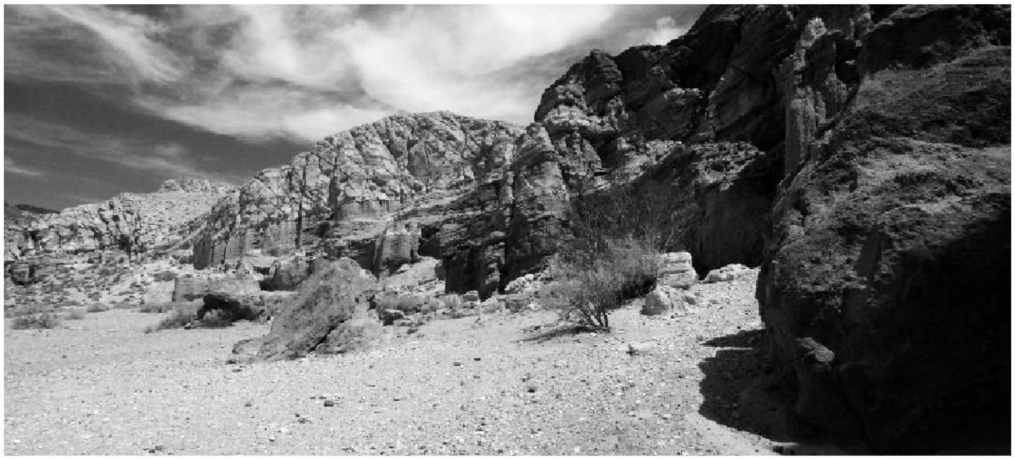




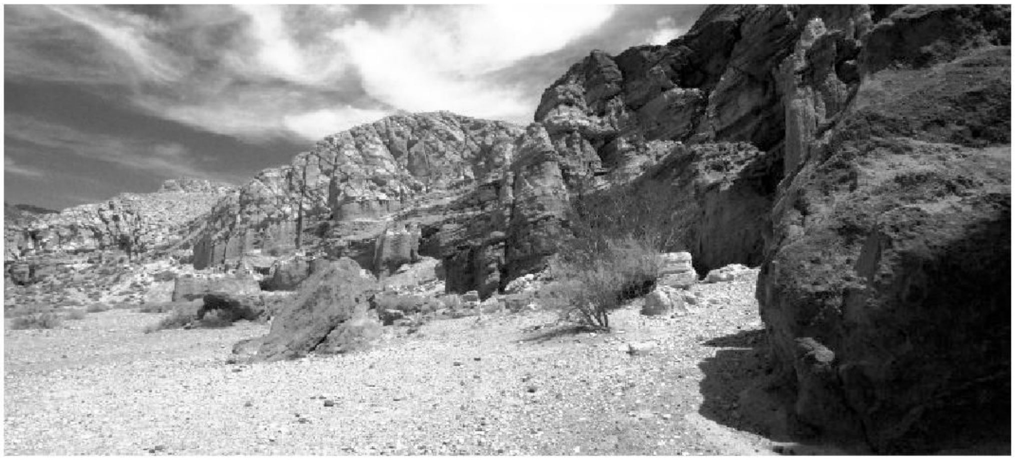
Q 7

Part I (q7a.m)

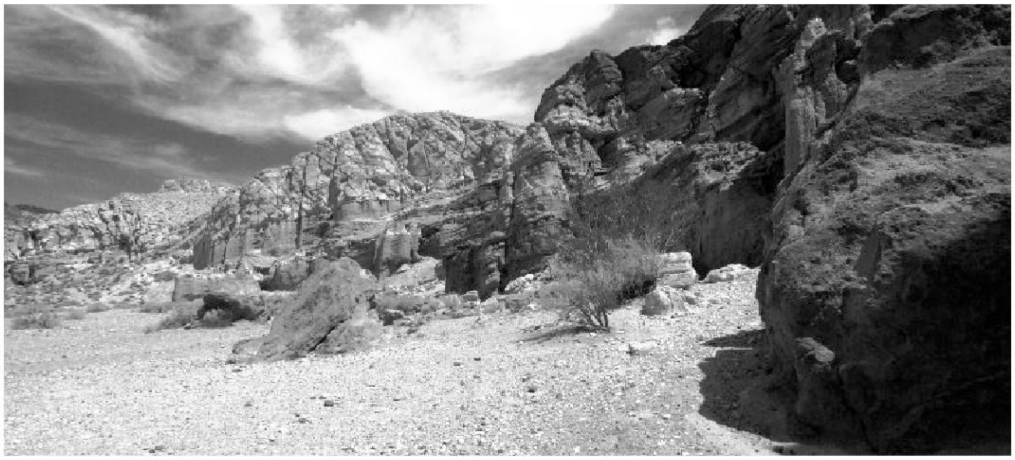
Original Image (*bw_canyon.jpg):*



Applying histogram Equalization once (*hist1_canyon.jpg)*:



Applying histogram Equalization twice (*hist2_canyon)*:

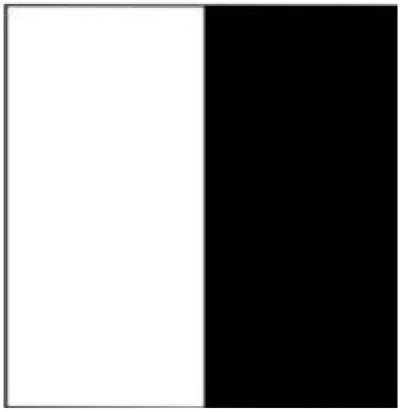


The last two images are similar. This implies that histogram equalization is idempotent and it doesn't affect the image if it's applied more than once.

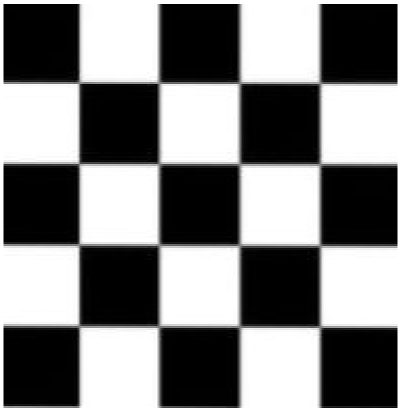
Part II (q7b.m)

1. Image with similar histograms show no change.

   Input images:

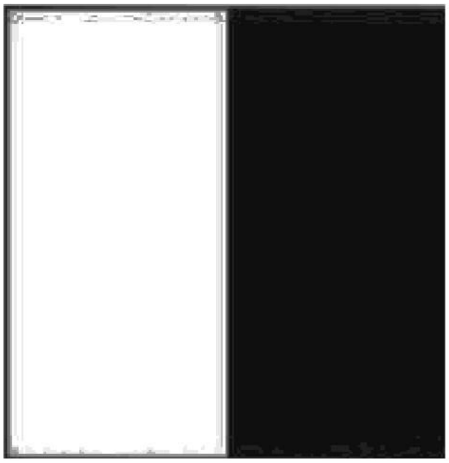


samehist1.png



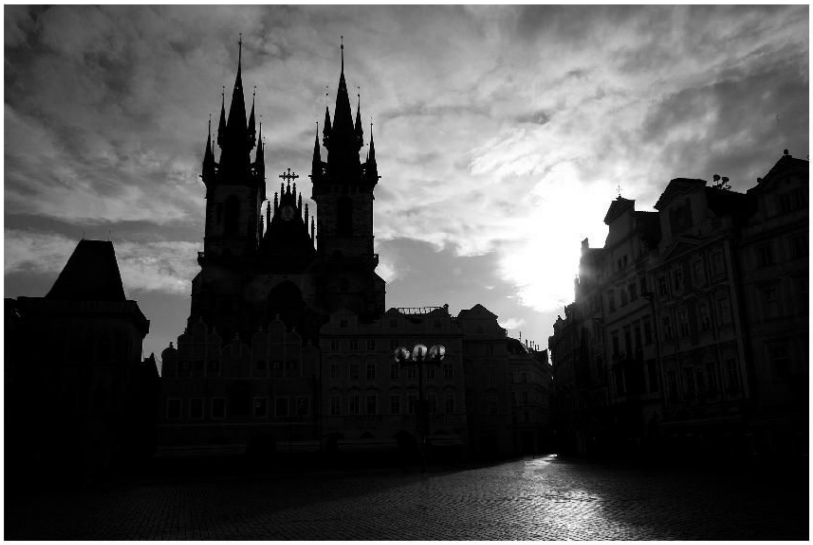
samehist2.png

Output:

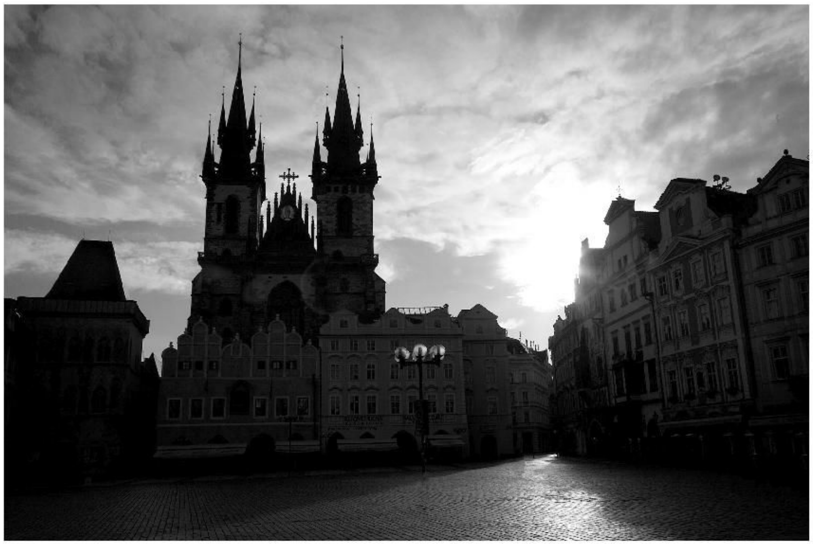


samehist_out.jpg

2. and 3. Input images:

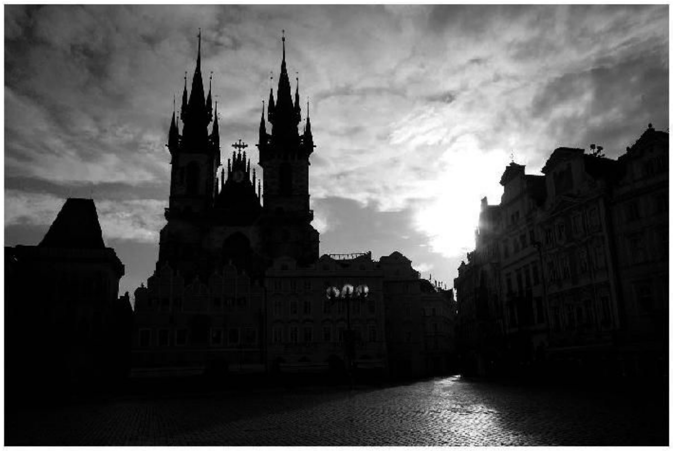


Dark Image
darkchurch.jpg

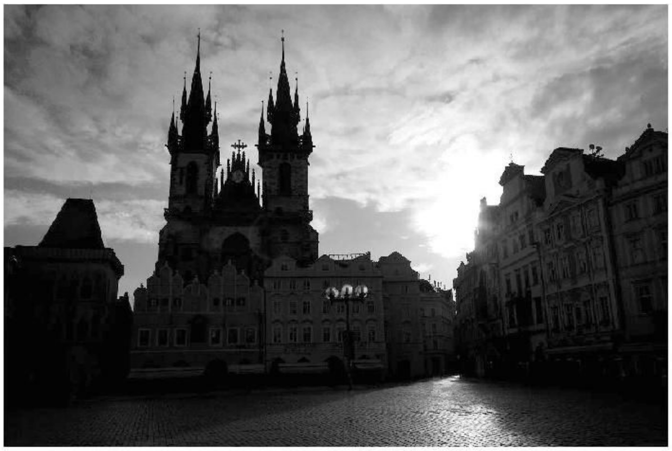


Light Image
lightchurch.jpg

Output Images:



Dark → Light
dark2light.jpg



Light → Dark
light2dark.jpg