RAMANUJAN COLLEGE
UNIVERSITY OF DELHI

DSE – 2 : Digital Image Processing
PRACTICAL FILE

SUBMITTED TO: Mrs. Bhavya Ahuja Ma'am
SUBMITTED BY: Sukaina Inam Naqvi
ROLL NO – 20201426
Examination Roll No.- 20020570033

B.Sc. (H) Computer Science | V Semester

Question 1 :

Write program to read and display digital image using MATLAB or SCILAB

a. Become familiar with SCILAB/MATLAB Basic commands

b. Read and display image in SCILAB/MATLAB

c. Resize given image

d. Convert given color image into gray-scale image

e. Convert given color/gray-scale image into black & white image

f. Draw image profile

g. Separate color image in three R G & B planes

h. Create color image using R, G and B three separate planes

i. Flow control and LOOP in SCILAB

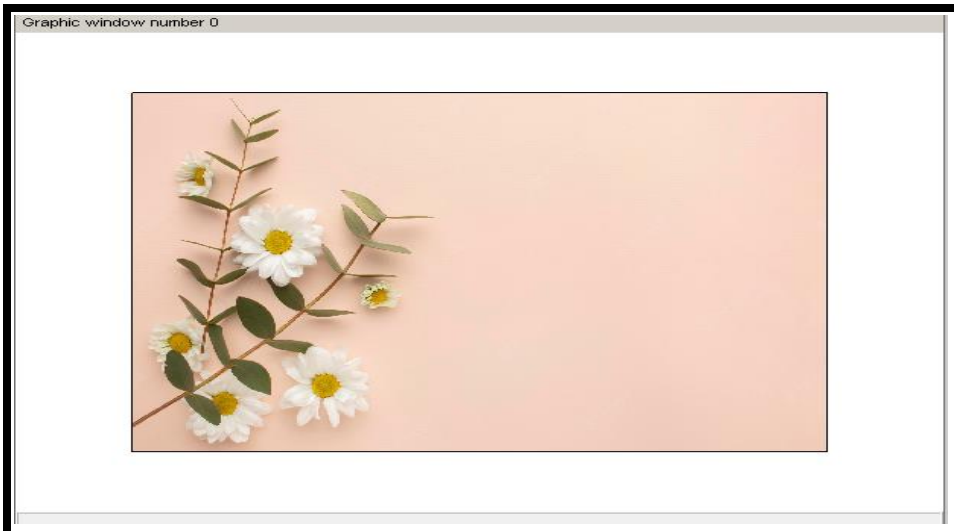
j. Write given 2-D data in image file

Solution :

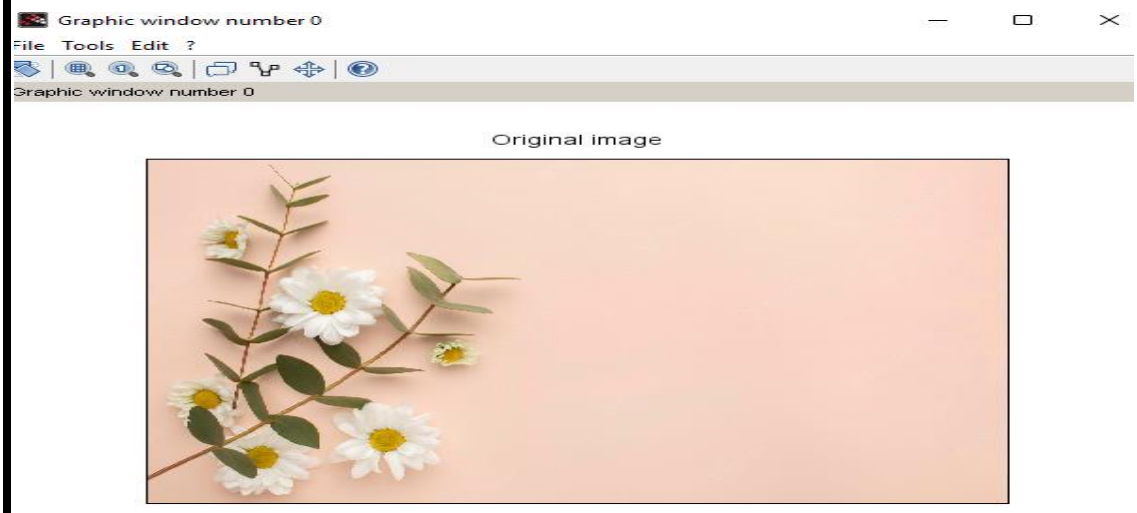
Code -

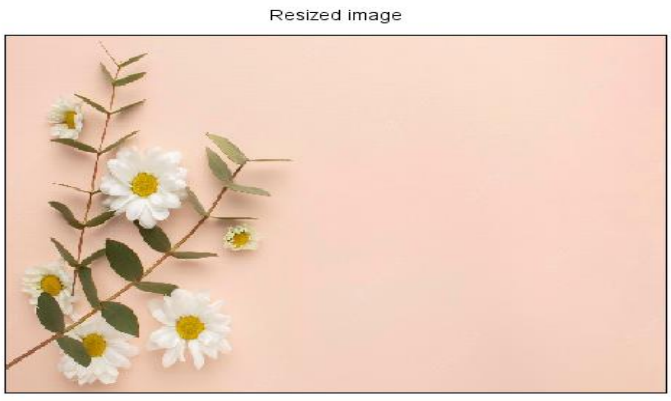
(a),(b)

```
--> // Question 1

--> // a. Becoming familiar with SCILAB Basic commands

--> // b.  Read and display image in SCILAB

--> I1=imread("img1.jpg");

--> imshow(I1);

-->
```




```
--> // c.Resize given image

--> I= imread("img1.jpg");

--> J= imresize(I,0.3);

--> imshow(I);

--> title("Original image");

--> imshow(J);

--> title("Resized image");
```

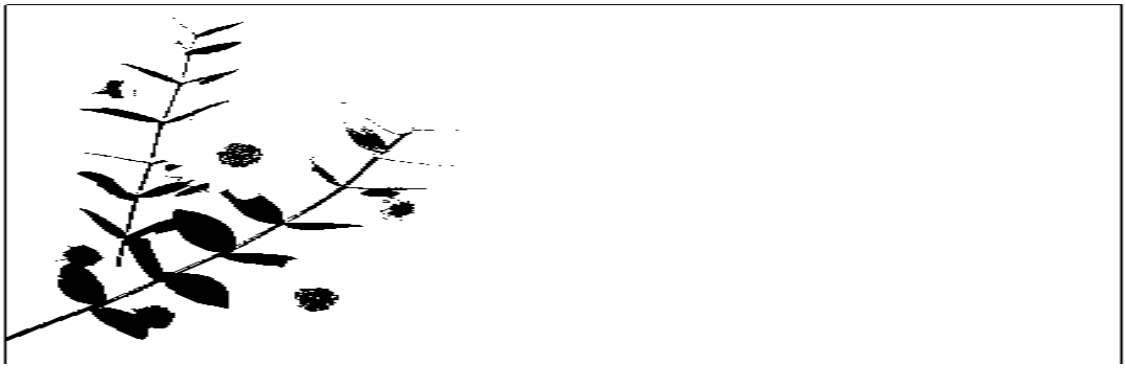




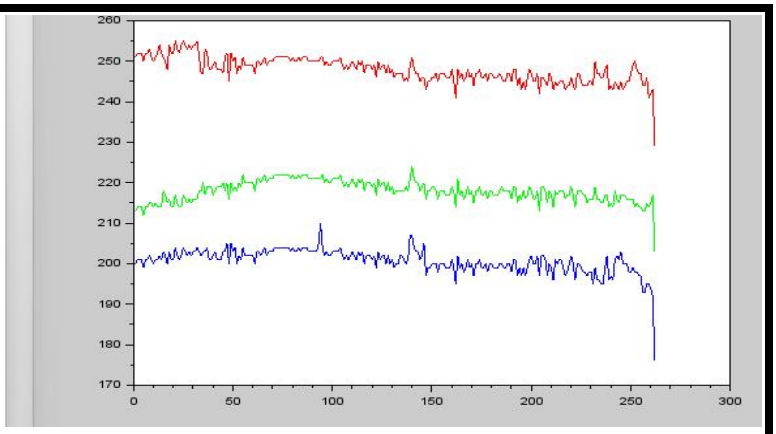
```
--> // d. Convert given color image into gray-scale image

--> Ilgray= rgb2gray(I1);

--> imshow(Ilgray);
```

Graphic window number 0

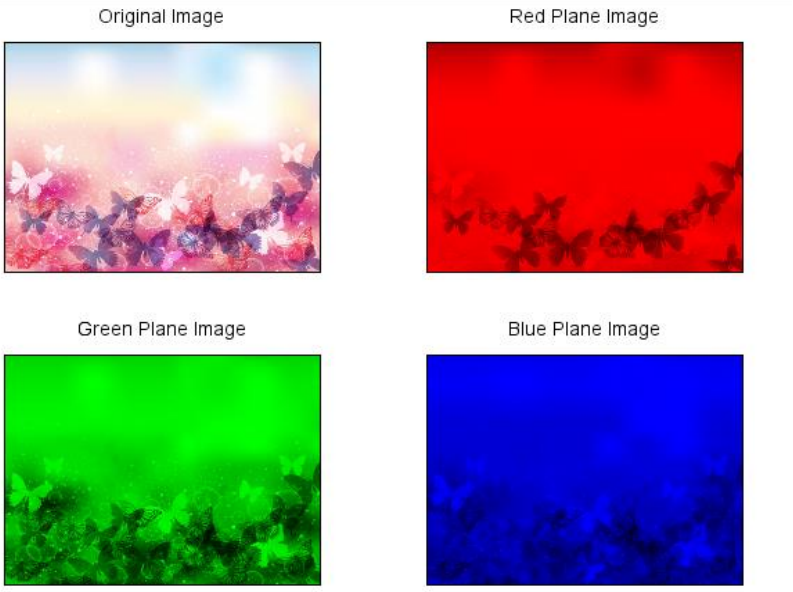
```
-> // e. Convert given color/gray-scale image into black & white image

-> I2= im2bw(Ilgray,0.5);

-> imshow(I2);
```



```
--> //f. Draw image profile

--> I=imread("img1.jpg");

--> improfile(I);
```

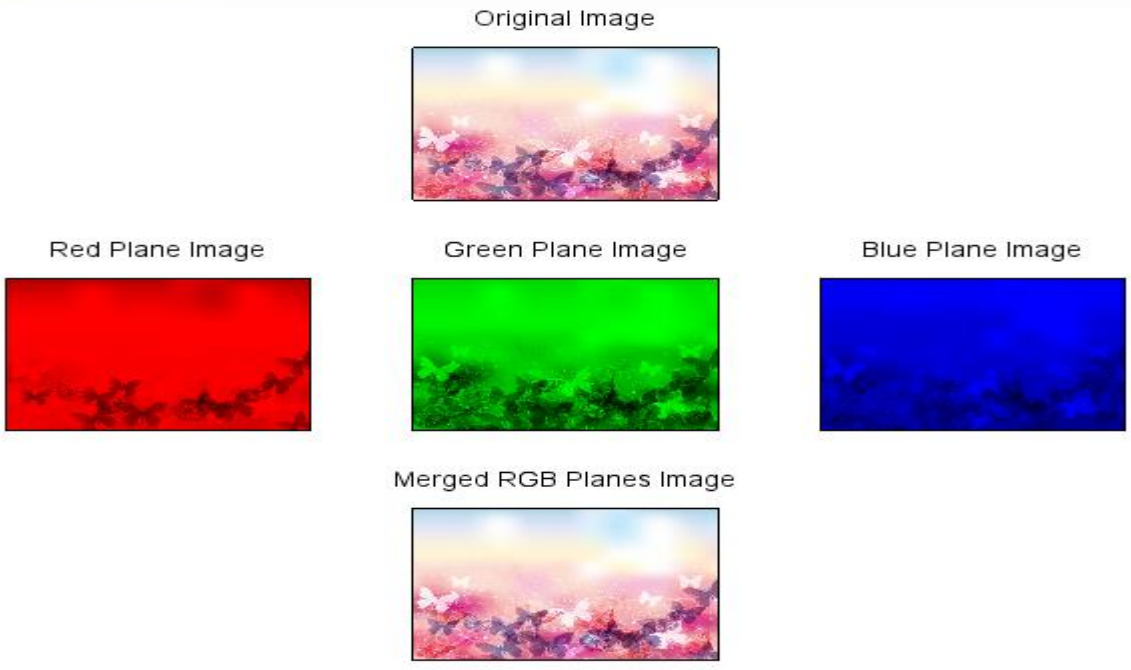



```
--> //g. Separate color image in three R G & B planes

--> img=imread("img4.jpg");

-->

--> [r,c] = size(img); // no. of rows & columns of image

-->

--> all_black = zeros(r,c,'uint8'); // A Black Image

-->

--> // img(:,:,1) = red channel of image

--> red_img = cat(3, img(:,:,1), all_black, all_black);

-->

--> // img(:,:,2) = green channel of image

--> green_img = cat(3,  all_black, img(:,:,2), all_black);

-->

--> // img(:,:,3) = blue channel of image

--> blue_img = cat(3, all_black, all_black, img(:,:,3));

--> // plotting

--> subplot(2,2,1),title("Original Image"),imshow(img);

--> subplot(2,2,2),title("Red Plane Image"),imshow(red_img);

--> subplot(2,2,3),title("Green Plane Image"),imshow(green_img);

--> subplot(2,2,4),title("Blue Plane Image"),imshow(blue_img);
```




```
--> // h. Create color image using R, G and B three separate planes

--> img=imread("img4.jpg");

--> // RGB Image = red + green + blue channels

--> merged = red_img + green_img + blue_img;

-->

--> subplot(3,3,2),title("Original Image"), imshow(img);

--> subplot(3,3,4),title("Red Plane Image"), imshow(red_img);

--> subplot(3,3,5),title("Green Plane Image"), imshow(green_img);

--> subplot(3,3,6),title("Blue Plane Image"), imshow(blue_img);

--> subplot(3,3,8),title("Merged RGB Planes Image"), imshow(merged);
```

Graphic window number 0

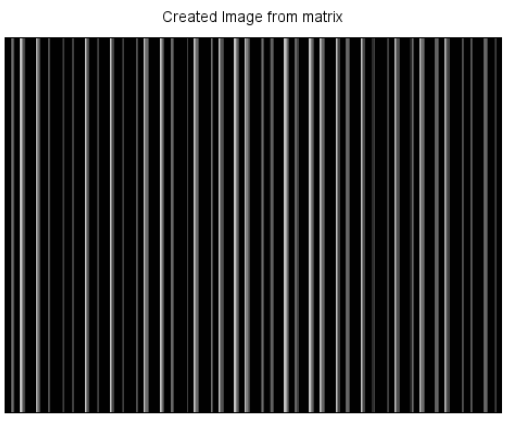


i.

```
num = input("Enter a Number  ");
% Flow Control
% Program to find a Number is even or Odd
if  mod(num,2) == 0
         disp('The Number is Even');
      elseif mod(num,2) ~=0
       disp('The Number is Odd');
       else
       disp('Invalid Number');
end
% Loop
% Program to generate  Bipolar signal  +1 / -1
 mat = rand(1,10,'single');
 binary =zeros(size(mat));
 for  count = 1:1:length(mat)
   if mat(count) >= 0
      binary(count) =1;
   else
      binary(count)  =-1;
   end
 end
 disp("Bipolar Signal ")
 disp(binary)
```

```
Enter a Number  4
The Number is Even
Bipolar Signal
     1     1     1     1     1     1     1     1     1     1
```

```
--> // j. Create image with given 2D data

--> mat = modulo(round(rand(20,20) * (17/7) * 128),256);

-->

--> disp('Given 2D data : ', mat);

  "Given 2D data : "

         column 1 to 13

   66.    96.    87.    120.   183.   176.   8.     45.    58.    215.   83.    26.    104.
   235.   34.    40.    31.    150.   178.   161.   158.   6.     238.   46.    104.   166.
   0.     67.    242.   39.    69.    254.   122.   163.   6.     111.   20.    37.    63.
   103.   97.    66.    107.   5.     18.    75.    174.   23.    239.   64.    32.    49.
   207.   112.   35.    117.   37.    174.   157.   175.   9.     170.   9.     226.   6.
   195.   91.    213.   228.   89.    39.    132.   146.   4.     30.    210.   24.    127.
   8.     176.   48.    81.    12.    226.   90.    242.   58.    41.    29.    123.   3.
   213.   150.   217.   155.   8.     83.    28.    246.   153.   69.    9.     176.   61.
   17.    103.   6.     82.    163.   170.   193.   49.    233.   4.     74.    220.   85.
   21.    184.   126.   163.   53.    51.    107.   255.   37.    255.   218.   211.   107.
   174.   156.   127.   167.   202.   230.   220.   132.   66.    41.    37.    128.   63.
   206.   136.   17.    37.    52.    1.     162.   77.    180.   45.    2.     44.    94.
   226.   84.    35.    70.    16.    183.   89.    31.    82.    204.   98.    154.   86.
   62.    197.   62.    195.   233.   96.    202.   31.    136.   76.    165.   130.   92.
   169.   126.   175.   237.   128.   79.    27.    145.   27.    164.   178.   12.    178.
   72.    30.    183.   15.    189.   194.   140.   123.   251.   7.     15.    89.    67.
   72.    14.    213.   209.   10.    36.    225.   11.    252.   245.   0.     78.    214.
   67.    150.   21.    63.    20.    190.   23.    161.   81.    39.    180.   105.   182.
   19.    82.    157.   122.   1.     211.   75.    3.     129.   245.   87.    122.   131.
   203.   129.   109.   2.     32.    103.   135.   190.   112.   107.   41.    146.   133.


         column 14 to 20

   21.    192.   25.    42.    76.    40.    150.
   109.   54.    137.   5.     16.    129.   50.
   148.   15.    3.     92.    152.   145.   139.
   239.   31.    225.   146.   119.   70.    72.
   23.    45.    136.   132.   29.    107.   69.
   183.   153.   96.    131.   22.    73.    249.
   74.    93.    16.    32.    228.   206.   241.
   127.   19.    166.   133.   242.   84.    150.
   134.   196.   96.    244.   243.   50.    244.
   12.    7.     104.   10.    67.    230.   48.
   199.   47.    73.    62.    106.   90.    240.
   128.   39.    80.    222.   134.   124.   22.
   5.     85.    9.     162.   195.   48.    38.
   0.     149.   150.   33.    41.    2.     217.
   158.   151.   189.   129.   141.   218.   41.
   148.   210.   51.    186.   1.     21.    15.
   43.    3.     49.    172.   3.     203.   165.
   49.    4.     103.   10.    47.    129.   92.
   24.    170.   112.   96.    239.   11.    217.
   28.    6.     86.    38.    199.   74.    35.

-->

--> title('Created Image from matrix');

--> imshow(mat2utfimg(mat));
```



Question 2 : To write and execute image processing programs using point processing method

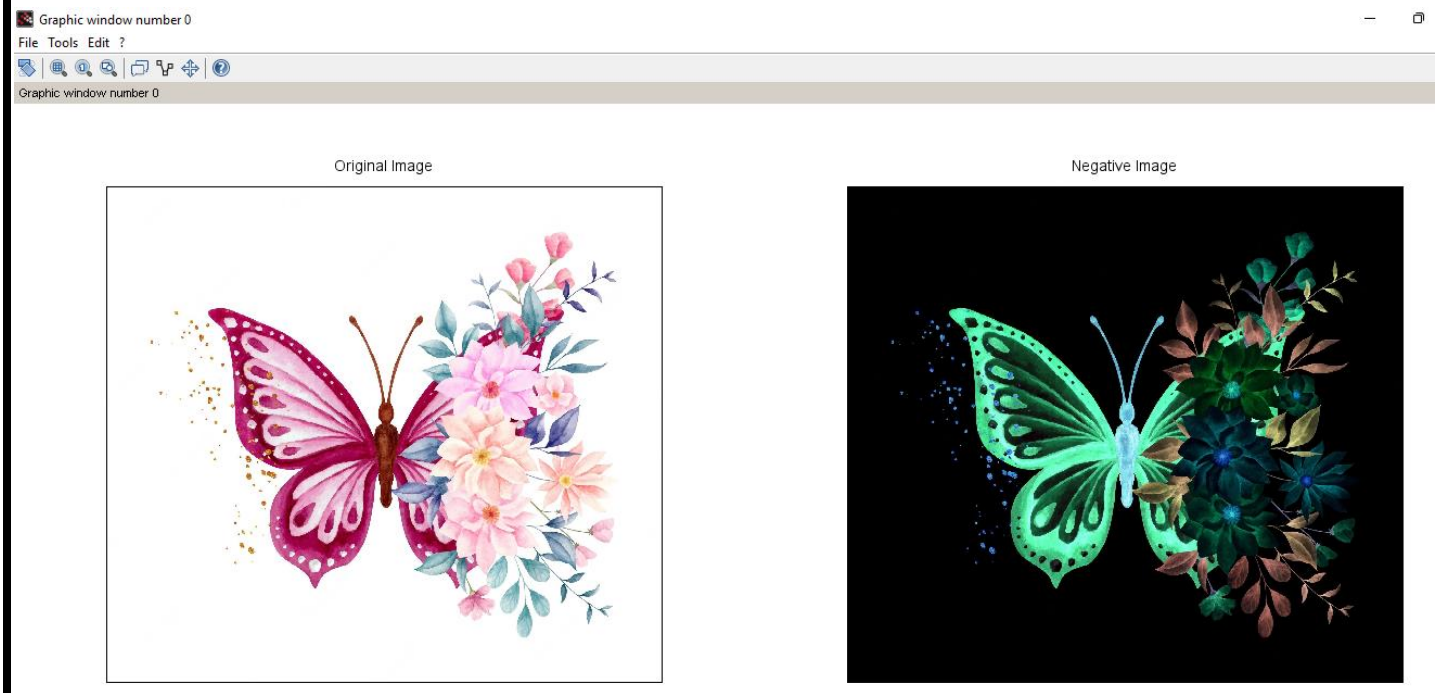
a. Obtain Negative image

b. Obtain Flip image

c. Thresholding

d. Contrast stretching

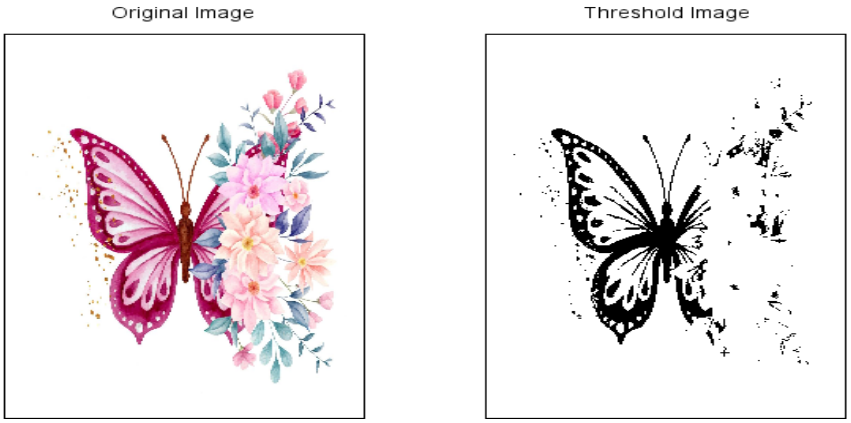
```
--> // 2. To write and execute image processing programs using point processing method

--> // a. Obtain Negative image

--> img=imread("img5.jpg");

--> subplot(1,2,1)

--> imshow(img);

--> title("Original Image");

--> L=2^8;

--> neg=(L-1)- img;

--> subplot(1,2,2)

--> imshow(neg);

--> title("Negative Image");
```



Scilab 6.1.1 Console

```
--> // b. Obtain Flip image
--> img=imread("img5.jpg");
--> subplot(1,2,1)
--> imshow(img);
--> title("Original Image");
--> Flip_img= imrotate(img,90);
--> subplot(1,2,2)
--> imshow(Flip_img);
--> title("Flip Image");
```


```
--> //c.  Thresholding
--> img=imread("img5.jpg");
--> subplot(1,2,1)
--> imshow(img);
--> title("Original Image");
--> th_img=im2bw(img,0.5);
--> subplot(1,2,2)
--> imshow(th_img);
--> title("Threshold Image");
```



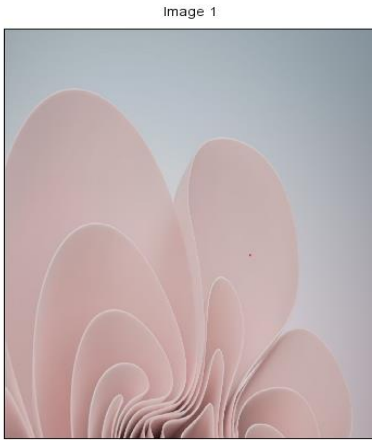
```
--> // (d)  Contrast stretching

--> I=imread("img5.jpg");

--> imshow(I);

--> title("Original Image");

--> J=imadjust(I,stretchlim(I),[]);

--> imshow(J);

--> title("Contrast Stretched Image");
```

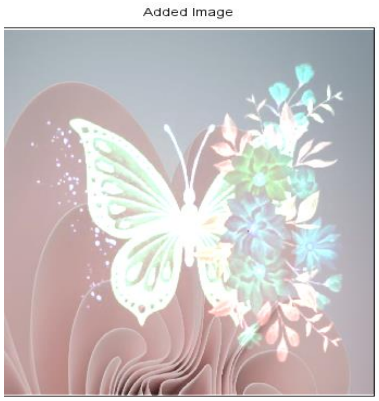
Contrast Stretched Image

Question 3-

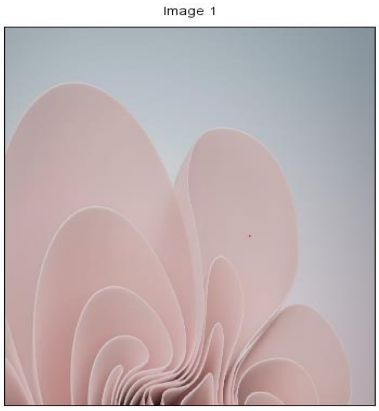
```
--> // 3. To write and execute programs for image arithmetic operations

--> // a. Addition of two images
```

```
--> img1=imread("img6.jpg");

--> subplot(1,3,1)

--> title("Image 1");

--> imshow(img1);

--> img2=imread("img7.jpg");

--> subplot(1,3,2)

--> title("Image 2");

--> imshow(img2);

--> added_img=imadd(img1,img2);

--> subplot(1,3,3)

--> title("Added Image");

--> imshow(added_img);
```






```
--> // b. Subtract one image from other image

--> img1=imread("img6.jpg");

--> subplot(1,3,1)

--> title("Image 1");

--> imshow(img1);

--> img2=imread("img7.jpg");

--> subplot(1,3,2)

--> title("Image 2");

--> imshow(img2);

--> sub_img=imsubtract(img1,img2);

--> subplot(1,3,3)

--> title("Subtracted Image");

--> imshow(sub_img);
```



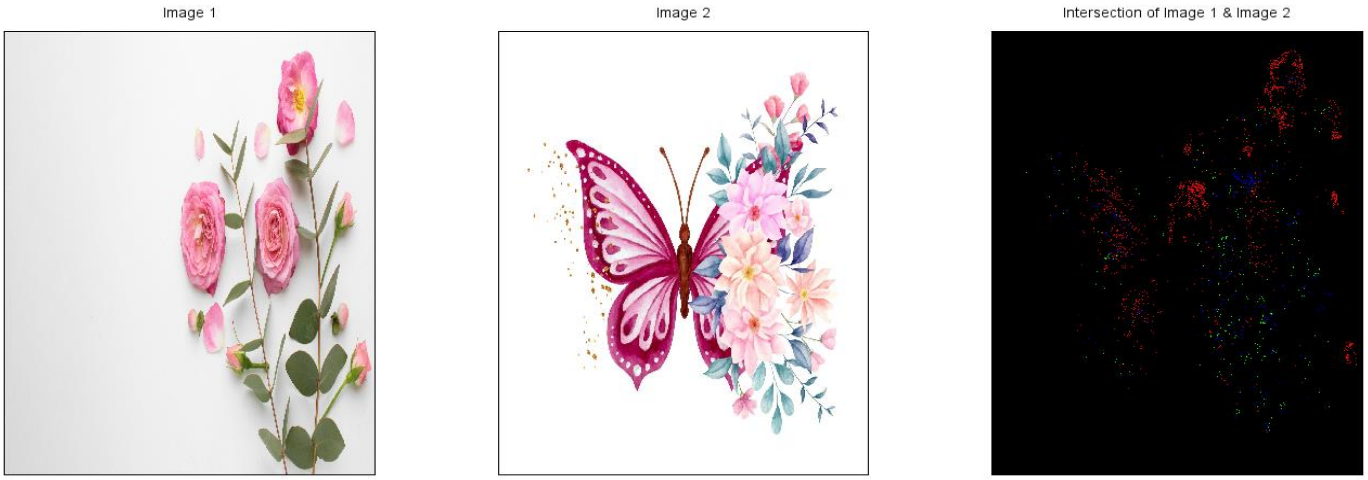

```
--> //c. Mean of the images

--> im1=imread("img1.jpg");

--> M=mean(im2double(im1));

--> disp("Mean of the image: "), disp(M);

  "Mean of the image: "

   0.8161858
```

QUESTION 4:-

```
--> // Question 4 :  To write and execute programs for image logical operations

--> // a. AND operation between two images

--> i= imread('img1.jpg');

--> j= imread('img6.jpg');

--> AND= bitand(i,j);

--> subplot(1,3,1)

--> imshow(i)

--> title("Image 1")

--> subplot(1,3,2)

--> imshow(j)

--> title("Image 2")

--> subplot(1,3,3)

--> imshow(AND)

--> title("AND Operation")
```


```
--> // b. OR operation between two images

--> i= imread('img1.jpg');

--> j= imread('img2.jpg');

--> OR= bitor(i,j);

--> subplot(1,3,1)

--> imshow(i)

--> title("Image 1")

--> subplot(1,3,2)

--> imshow(j)

--> title("Image 2")

--> subplot(1,3,3)

--> imshow(OR)

--> title("OR Operation")
```


```
--> // c.  Calculate intersection of two image

--> img1 = imread('img2.jpg');

--> img2 = imread('img5.jpg');

--> subplot(1,3,1);

--> title('Image 1');

--> imshow(img1);

--> subplot(1,3,2);

--> title('Image 2');

--> imshow(img2);

--> inter_img = (double(img1) - double(img2)) == 0;

--> subplot(1,3,3)

--> title('Intersection of Image 1 & Image 2');

--> imshow(inter_img);
```




```
--> //d. NOT operation (Negative image)

--> W= imread('img1.jpg');

--> NotW= bitcmp(W);

--> subplot(1,2,1)

--> imshow(W)

--> title("Image 1")

--> subplot(1,2,2)

--> imshow(NotW)

--> title("NOT Operation")
```

.

Image 1

NOT Operation

Q. 5) To write a program for histogram calculation and equalization using :

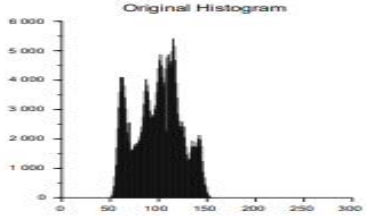
a. Standard MATLAB function

b. Program without using standard MATLAB functions

```
/* a. With standard Matlab function */
lena_img = imread('Test_images/Lena_dark.png');

subplot(1,2,1),title("Original Image "),imshow(lena_img);
subplot(1,2,2),title("Original Histogram"),imhist(lena_img,[],1);
```

Original Image

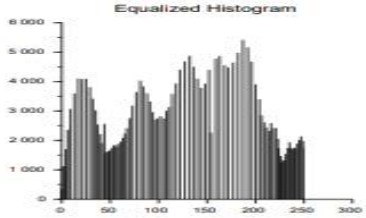
Original Histogram




```
In [183]:
h_img = imhistequal(lena_img);

subplot(1,2,1),title("Equalized Image "),imshow(h_img);
subplot(1,2,2),title("Equalized Histogram"),imhist(h_img,[],1);
```

Equalized Image

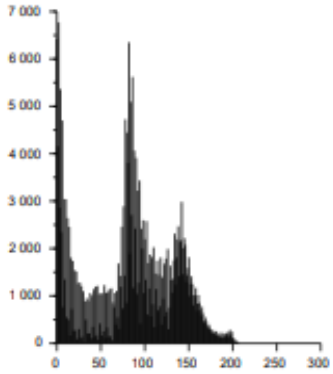
Equalized Histogram



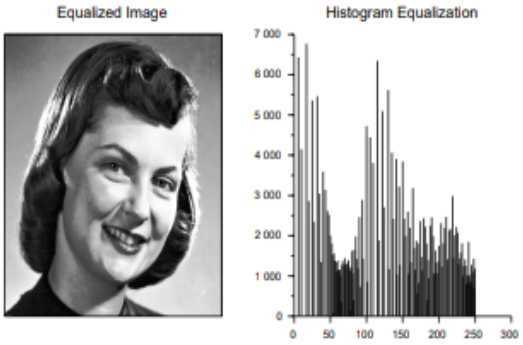

```
/* b. Without standard Matlab function */

g_img = imread('Test_images/girlface.bmp');

subplot(1,2,1),title("Original Image "),imshow(g_img);
subplot(1,2,2),title("Original Histogram"),imhist(g_img,[],1);
```

Original Image

Original Histogram



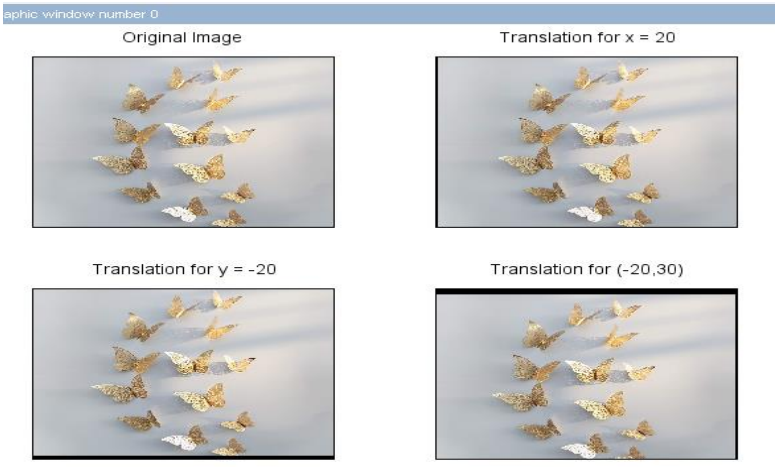
```
/* Algorithm */

function eq_img = histeq(g_img)
    [freq, bins] = imhist(g_img,256);
    bins = 255;
    [mr, nc] = size(g_img);
    freq = cumsum(freq);
    npixels = prod(size(g_img));
    output = round(bins.*(freq./npixels));

    // Creating Equalized Image
    for i = 1:mr
        for j = 1:nc
            eq_img(i,j) = output(g_img(i,j) + 1);
        end
    end
endfunction

he_img = uint8(histeq(g_img));
subplot(1,2,1),title("Equalized Image "),imshow(he_img);
subplot(1,2,2),title("Histogram Equalization"),imhist(he_img, [], 1);
```




```
--> // Question 6. To write and execute program for geometric transformation of image

--> // a. Translation

--> S1 = imread('img3.jpg');

-->

--> // Translation for x  = 20

-->

--> mat = [  1 0 0;...
  >          0 1 0;...
  >         20 0 1];

--> S2 = imtransform(S1,mat,'affine');

-->

--> // Translation for y  = -20

-->

--> mat = [  1 0 0;...
  >          0 1 0;...
  >          0 -20 1];

--> S3 = imtransform(S1,mat,'affine');
```

```
--> mat = [  1 0 0;...
  >          0 1 0;...
  >        -20 30 1];

--> S4 = imtransform(S1,mat,'affine');

-->

--> subplot(2,2,1);

--> title('Original Image');

--> imshow(S1);

--> subplot(2,2,2);

--> title('Translation for x = 20');

--> imshow(S2);

--> subplot(2,2,3);

--> title('Translation for y = -20');

--> imshow(S3);

--> subplot(2,2,4);

--> title('Translation for (-20,30)');

--> imshow(S4);
```

aphic window number 0




```
--> // b. Scaling

-->

--> s_img = imread('img3.jpg');

--> width = size(s_img, 'c'); // column pixels = width

--> height = size(s_img, 'r'); // row pixels = height

--> // Scaling width by 2

--> w = 2;

--> h = 1;

--> mat = [ w 0;
  >         0 h;
  >         0 0];

--> scl = imtransform(s_img, mat, 'affine', width*w, height*h);

--> // Scaling height by 2

--> w = 1;

--> h = 2;

--> mat = [ w 0;
  >         0 h;
  >         0 0];
```

```
--> sc2 = imtransform(s_img, mat, 'affine', width*w, height*h);

--> // Scaling image by 2

--> w = 2;

--> h = 2;

--> mat = [ w 0;
  >         0 h;
  >         0 0];

--> sc3 = imtransform(s_img, mat, 'affine', width*w, height*h);

--> function s = str(img)
  >     s = 'Size : ' + strcat(string(size(img)), ' * ');
  > endfunction;

--> subplot(3,3,1);

--> title('Original Image');

--> xlabel(str(s_img));

--> imshow(s_img);

--> subplot(3,2,2);

--> title('Image scaling width by 2');

--> xlabel(str(sc1));

--> imshow(sc1);
--> subplot(2,3,4);

--> title('Image scaling height by 2');

--> xlabel(str(sc2));

--> imshow(sc2);

--> subplot(2,2,4);

--> title('Image scaling by 2');

--> xlabel(str(sc3));

--> imshow(sc3);
```

Original Image

Size : 1002 * 2000 * 3

Image scaling width by 2

Size : 1002 * 4000 * 3

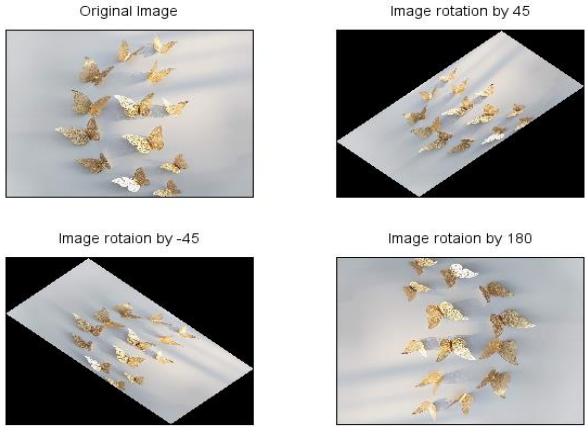
Image scaling height by 2

Size : 2004 * 2000 * 3

Image scaling by 2

Size : 2004 * 4000 * 3

```
--> // c. Rotation

-->

--> s_img = imread('img3.jpg');

--> subplot(2,2,1);

--> title('Original Image');

--> imshow(s_img);

--> subplot(2,2,2);

--> title('Image rotation by 45');

--> imshow(imrotate(s_img, 45));

--> subplot(2,2,3);

--> title('Image rotaion by -45');

--> imshow(imrotate(s_img, -45));

--> subplot(2,2,4);

--> title('Image rotaion by 180');

--> imshow(imrotate(s_img, 180));
```




```
--> //Shrinking

--> s_img = imread('img3.jpg');

--> im_50 = imresize(s_img, 0.5);

--> im_sw = imresize(s_img, [height, 100]);

--> im_sh = imresize(s_img, [150, width]);

-->

--> subplot(2,2,1);

--> title('Original Image');

--> xlabel(str(s_img));

--> imshow(s_img);

--> subplot(2,3,3);

--> title('Image with Shrinked Width');

--> xlabel(str(im_sw));

--> imshow(im_sw);

--> subplot(3,2,5);

--> title('Image with Shrinked Height');

--> xlabel(str(im_sh));

--> imshow(im_sh);

--> subplot(3,3,9);

--> title('Image Shrinked by 50%');

--> xlabel(str(im_50));

--> imshow(im_50);
```

Original Image

Size : 1002 * 2000 * 3

Image with Shrinked Width

Size : 1002 * 100 * 3

Image with Shrinked Height

Size : 150 * 2000 * 3

Image Shrinked by 50%

Size : 501 * 1000 * 3

```
--> // e. Zooming

--> iml=imread("img3.jpg");

--> title("Original image");

--> imshow(iml);

--> zoom_rect();

--> title("Zoomed image");
```

Zoomed image

Q.7) To understand various image noise models and to write programs for :

a. image restoration

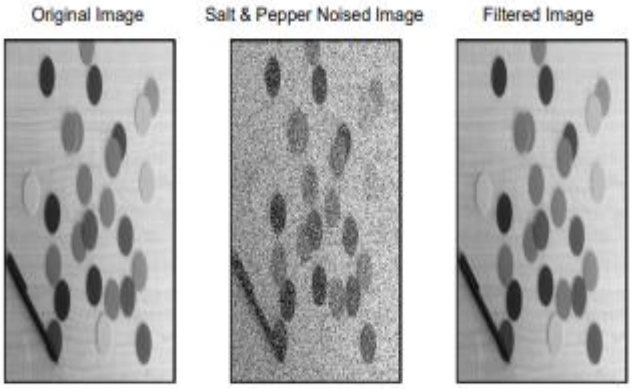
b. Remove Salt and Pepper Noise

c. Minimize Gaussian noise

d. Median filter

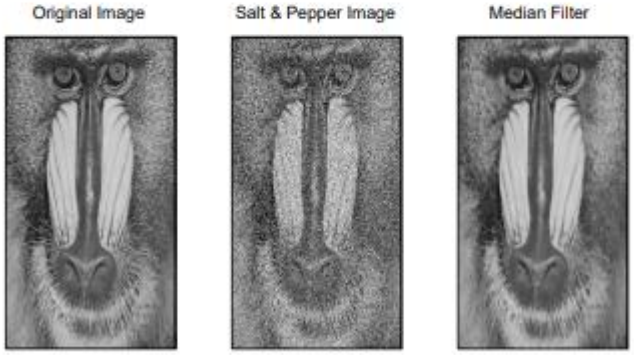
```
/* a. Image Restoration */

im1 = imread('Test_images/Kodim17_noisy.jpg');
f = fspecial('gaussian', [8, 8], 2);

subplot(121), title('Noisy Image'), imshow(im1);
subplot(122), title('Filtered Image'), imshow(imfilter(im1, f));
```


```
/* b. remove salt & pepper noise */

im2 = rgb2gray(imread('Test_images/coloredChips.png'));
im3 = imnoise(im2, 'salt & pepper', 0.3);

subplot(131), title('Original Image'), imshow(im2);
subplot(132), title('Salt & Pepper Noised Image'), imshow(im3);
subplot(133), title('Filtered Image'), imshow(immedian(im2,3));
```



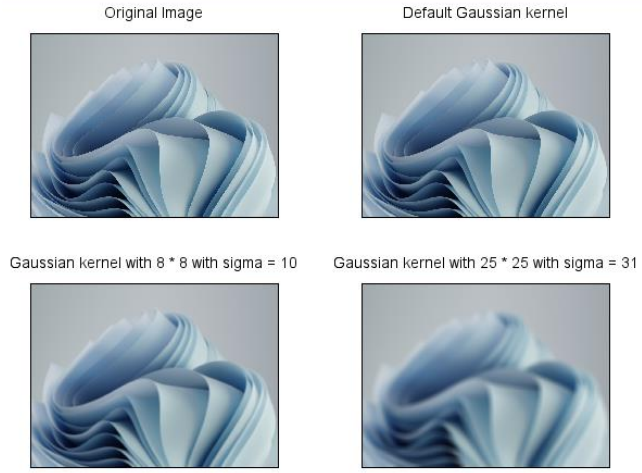

```
/* c. Minimize Gaussian Noise */

im1 = imread('Test_images/lena.jpeg');
im2 = imnoise(im1, 'gaussian');
f = fspecial('average', 3);

subplot(131), title('Original Image'), imshow(im1);
subplot(132), title('Gaussian Noised Image'), imshow(im2);
subplot(133), title('Filtered Image'), imshow(imfilter(im1, f));
```

```
/* d. Median Filter */

im2 = rgb2gray(imread(fullpath(getIPCVpath() + 'images/baboon.png')));
d_im = imnoise(im2, 'salt & pepper', 0.25);

[r c] = size(d_im);
img1 = zeros(r+2, c+2, 'uint8');
img1(2:r+1, 2:c+1) = d_im(:,:);

// border padded image
img1(1, 1) = d_im(1, 1);
img1(r+2, 1) = d_im(r, 1);
img1(1, c+2) = d_im(1, c);
img1(r+2, c+2) = d_im(r, c);

img1(2:r+1, 1) = d_im(:,1);
img1(2:r+1, c+2) = d_im(:,c);
img1(1, 2:c+1) = d_im(1,:);
img1(r+2, 2:c+1) = d_im(r,:);

for i = 2:r+1
    for j = 2:c+1
            img1(i,j) = gsort(img1(i-1:i+1, j-1:j+1))(5);
    end
end

subplot(131), title('Original Image'), imshow(im2);
subplot(132), title('Salt & Pepper Image'), imshow(d_im);
subplot(133), title("Median Filter"), imshow(img1(2:r+1, 2:c+1));
```

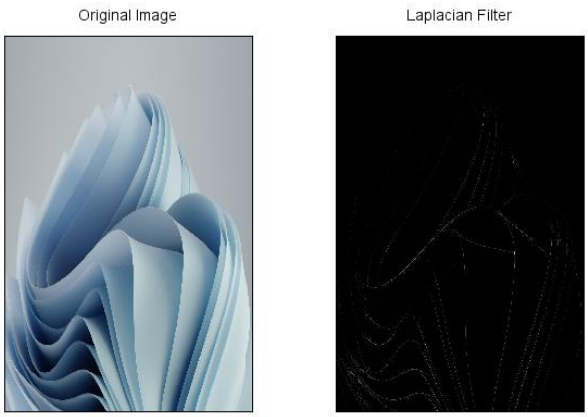



```
--> // Q. 8 ) Write and execute programs to use spatial low pass and high pass filters.

--> //Spatial Low Pass Filter

-->

--> i1 = imread('img8.jpg');

-->

--> g_filter = fspecial('gaussian');

--> i2 = imfilter(i1, g_filter);

-->

--> g_filter2 = fspecial('gaussian', [8,8], 10);

--> i3 = imfilter(i1, g_filter2);

-->

--> g_filter3 = fspecial('gaussian',[25,25], 31);

--> i4 = imfilter(i1, g_filter3);

-->

--> subplot(2,2,1);

--> title('Original Image');

--> imshow(i1);
```

```
--> subplot(2,2,2);

--> title('Default Gaussian kernel');

--> imshow(i2);

--> subplot(2,2,3);

--> title('Gaussian kernel with 8 * 8 with sigma = 10');

--> imshow(i3);

--> subplot(2,2,4);

--> title('Gaussian kernel with 25 * 25 with sigma = 31');

--> imshow(i4);
```



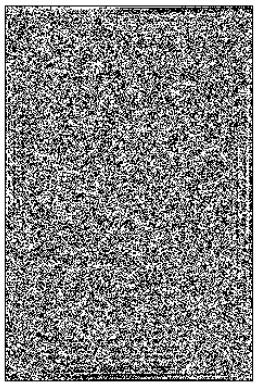

```
--> // Spatial High Pass Filter

-->

--> i1 = imread('img8.jpg');

--> l_filter = fspecial('laplacian');

--> i2 = imfilter(i1, l_filter);

--> subplot(1,2,1);

--> title('Original Image');

--> imshow(i1);

--> subplot(1,2,2);

--> title('Laplacian Filter');

--> imshow(i2);
```



```
--> //Q. 9 ) Write and execute programs for image frequency domain filtering :

--> //a. Apply FFT on given image

--> img = rgb2gray(imread('img4.jpg'));

--> ft_img = fft(double(img));

-->

--> subplot(1,2,1);

--> title('Original Image');

--> imshow(img);

--> subplot(1,2,2);

--> title('Direct Fourier Transformed Image');

--> imshow(ft_img);
```

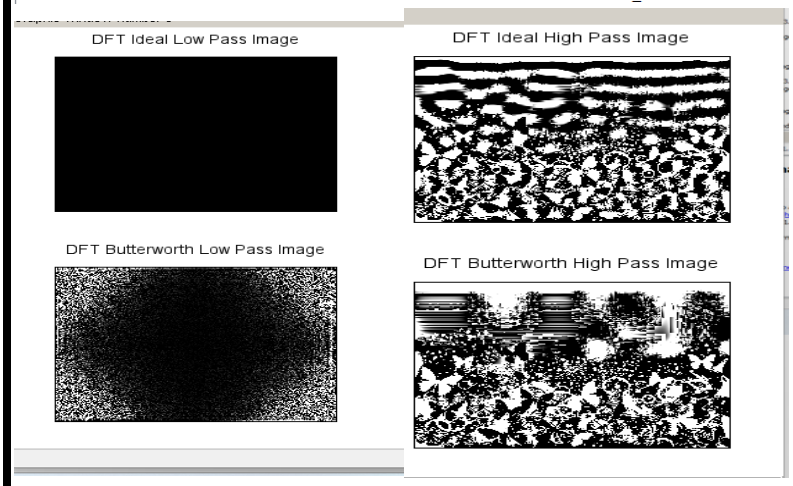
Original Image

Direct Fourier Transformed Image



```
--> // b. Perform low pass and high pass filtering in frequency domain

--> img = rgb2gray(imread('img4.jpg'));

--> [M,N] = size(ft_img);

--> D0 = 10;

--> u = 0:(M-1);

--> idx = find(u>M/2);

--> u(idx) = u(idx)-M;

--> v = 0:(N-1);

--> idy = find(v>N/2);

--> v(idy) = v(idy)-N;

--> [V,U] = meshgrid(v,u);

--> D = sqrt(U.^2 + V.^2);

--> // Ideal Filters

--> H11 = double(D <= D0); //low pass filter

--> G11 = H11.*ft_img;

--> H12 = 1.-H11; //high pass filter

--> G12 = H12.*ft_img;
```

```
--> // Butterworth Filters

--> o = 1; // filter order

--> H21 = 1./(1 + (D./D0).^(2*o)); //low pass filter

--> G21 = H21.*ft_img;

--> H22 = 1.-H21; //high pass filter

--> G22 = H22.*ft_img;

-->

-->

--> idl_l = ifft(G11);

--> idl_h = ifft(G12);

--> bwh_l = ifft(G21);

--> bwh_h = ifft(G22);

-->

--> subplot(2,2,1), title('DFT Ideal Low Pass Image'), imshow(idl_l);

--> subplot(2,2,2), title('DFT Ideal High Pass Image'),imshow(idl_h);

--> subplot(2,2,3), title('DFT Butterworth Low Pass Image'), imshow(bwh_l);

--> subplot(2,2,4), title('DFT Butterworth High Pass Image'),imshow(bwh_h);
```



Q. 10) Write a program in C and MATLAB/SCILAB for edge detection using different edge detection mask.

```
img = imread("Test_images\Lena_dark.png");
title("Original Image"), imshow(img);
```

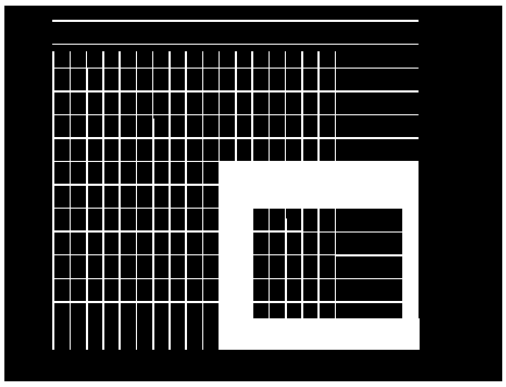

```
In [170]:
sobel = edge(img); // 0.2(Default)

sobel1 = edge(img, thresh = 0.5);

sobel2 = edge(img, thresh = -1);

title("Sobel Masks with threshold = 0.2, 0.5, -1");
imshow([sobel sobel1 sobel2]);
```

```
pre = edge(img, 'prewitt');

pre1 = edge(img, 'prewitt', thresh = 0.5);

pre2 = edge(img, 'prewitt', thresh = -1);

title("Prewitt Masks with threshold = 0.2, 0.5, -1");
imshow([pre pre1 pre2]);
```

Prewitt Masks with threshold = 0.2, 0.5, -1

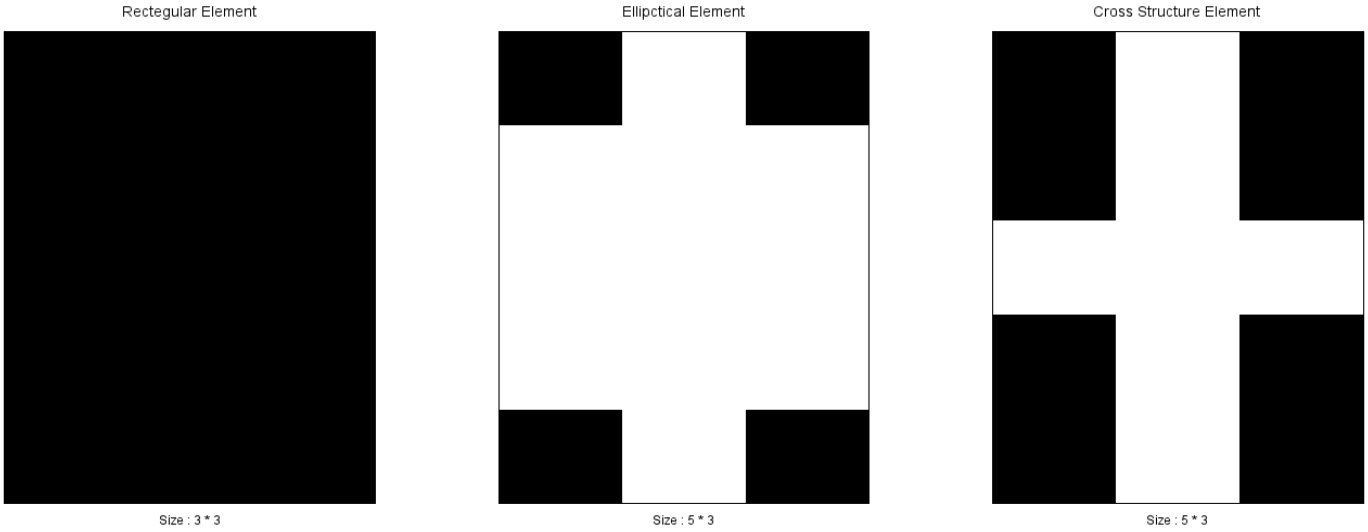
```
--> // Q. 11 ) Write and execute program for image morphological operations erosion and dilation.

--> // Morphological Operation : Erosion & Dilation

-->

--> function s = str(img)
  >     s = 'Size : ' + strcat(string(size(img)), ' * ');
  > endfunction;

-->

--> // Image

--> c = im2bw(imread('img7.jpg'), 0.5);

--> c = imcrop(c,[10,30,300,240]);

--> c(100:220, 130:250) = 1; // box

--> c(130:200, 150:240) = 0;

--> c(10:15:200, 30:250) = 1;

--> c(30:220, 30:10:200) = 1;

--> title('Original Image'), xlabel(str(c)), imshow(c);
```

Original Image



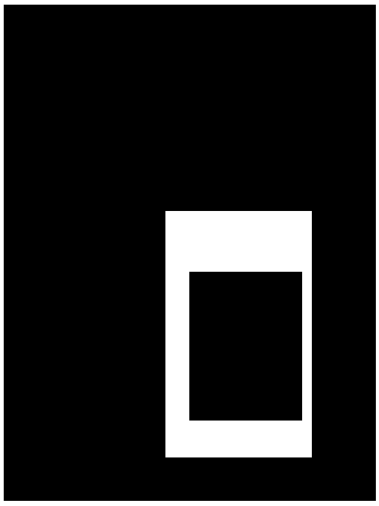
Size : 240 * 300

```
--> // Structure element

--> s1 = imcreatese('rect', 3, 3);

--> s2 = imcreatese('ellipse', 5, 3);

--> s3 = imcreatese('cross', 5, 3);

-->

--> // Plotting

--> subplot(1,3,1), title('Rectegular Element'), xlabel(str(s1)), imshow(mat2gray(s1));

--> subplot(1,3,2), title('Ellipctical Element'), xlabel(str(s2)), imshow(mat2gray(s2));

--> subplot(1,3,3), title('Cross Structure Element'), xlabel(str(s3)), imshow(mat2gray(s3));
```

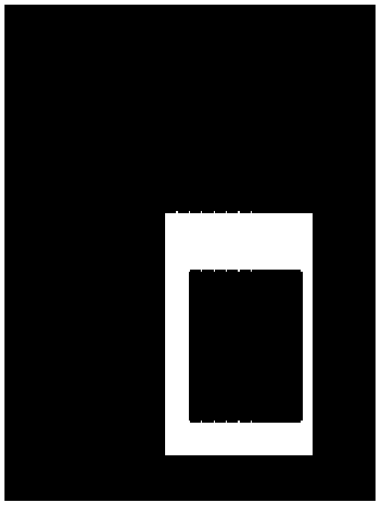



```
--> // erosion

--> e1 = imerode(c, s1);

--> e2 = imerode(c, s2);

--> e3 = imerode(c, s3);

-->

--> // Plotting

--> subplot(1,3,1), title('Rectegular Erosion'), imshow(e1);

--> subplot(1,3,2), title('Ellipctical Erosion'), imshow(e2);

--> subplot(1,3,3), title('Cross Structure Erosion'), imshow(e3);

-->
```

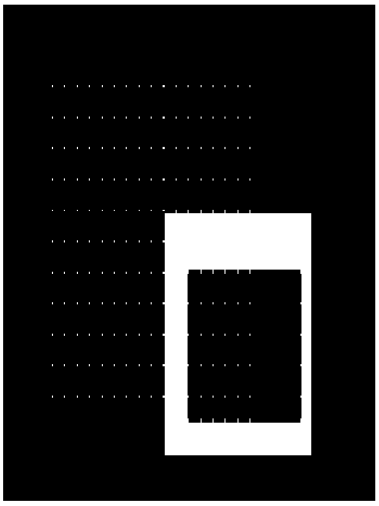
Rectegular Erosion



Ellipctical Erosion

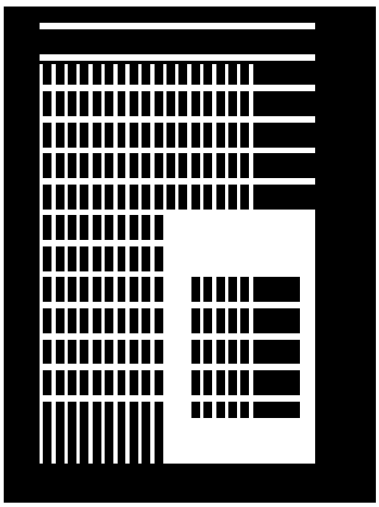


Cross Structure Erosion

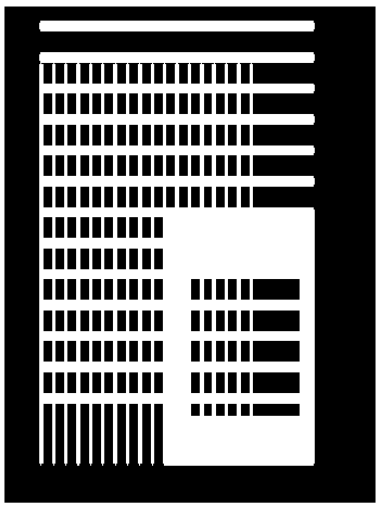


```
--> // dilation

--> d1 = imdilate(c, s1);

--> d2 = imdilate(c, s2);

--> d3 = imdilate(c, s3);

-->

--> // Plotting

--> subplot(1,3,1), title('Rectegular Dilation'), imshow(d1);

--> subplot(1,3,2), title('Ellipctical Dilation'), imshow(d2);

--> subplot(1,3,3), title('Cross Structure Dilation'), imshow(d3);

-->
```

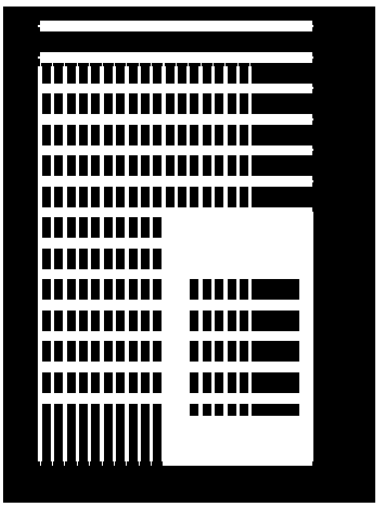
Rectegular Dilation



Ellipctical Dilation



Cross Structure Dilation



THANK YOU