Digital Image Processing

Name : Vimal Joshi

Course : Bsc (H) Computer Science

Exam Roll no. : 20020570040

Submitted to : Mrs. Bhavya Ahuja

Practicals

Q. 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

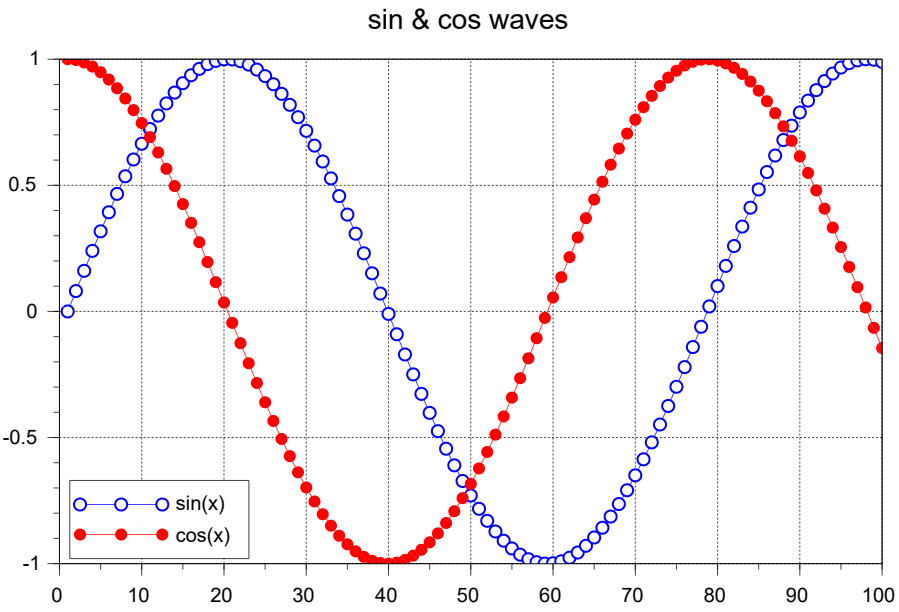
j. Write given 2-D data in image file

Code)

In [194]:

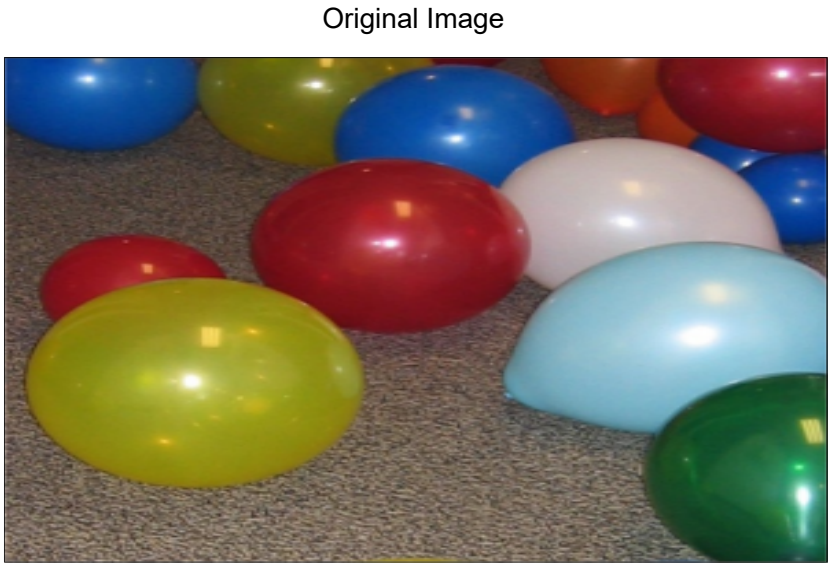
```
disp('2 * 3 - 4 + 8 / 3 \ 9 =', 2*3-4+8/3\9);
x = linspace(0,8,100);
plot(sin(x),'o-');
plot(cos(x),'r.-');
xtitle('sin & cos waves');
legend('sin(x)','cos(x)', 3);
xgrid(0,1,7);
```

```
"2 * 3 - 4 + 8 / 3 \ 9 ="

 5.375
```



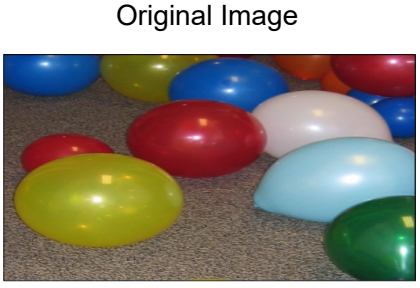
In [195]:

```
/* b. Reading & Displaying Image in Scilab */

img = imread("Test_images/balloons.png");
title('Original Image'),imshow(img);
```



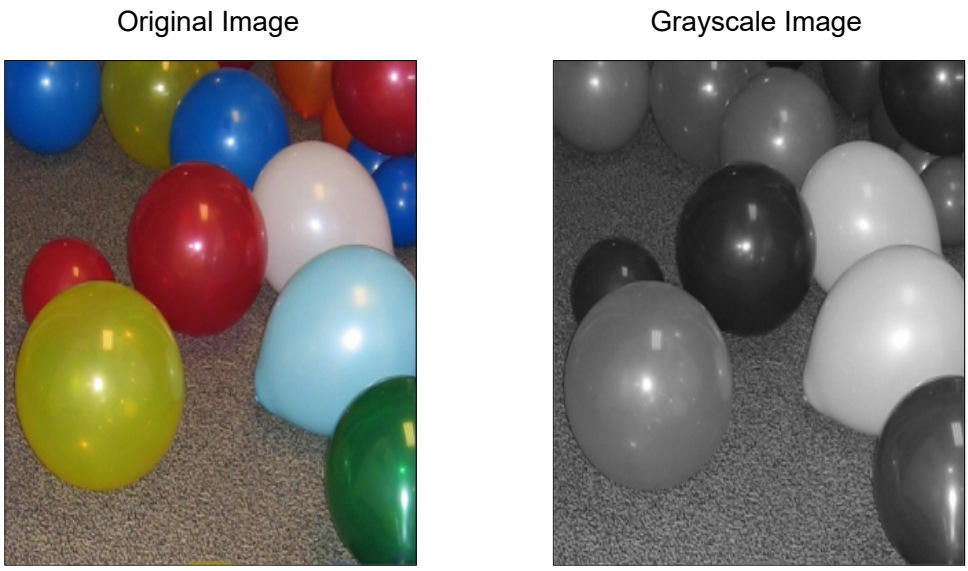
In [196]:

```
/* c. Resize given image */

img1 = imresize(img,2); // resized image

subplot(2,2,1), title('Original Image'), imshow(img);
xlabel(strcat(string(size(img)),' * '));

subplot(1,2,2), title('Resized Image'), imshow(img1);
xlabel(strcat(string(size(img1)),' * '));
```



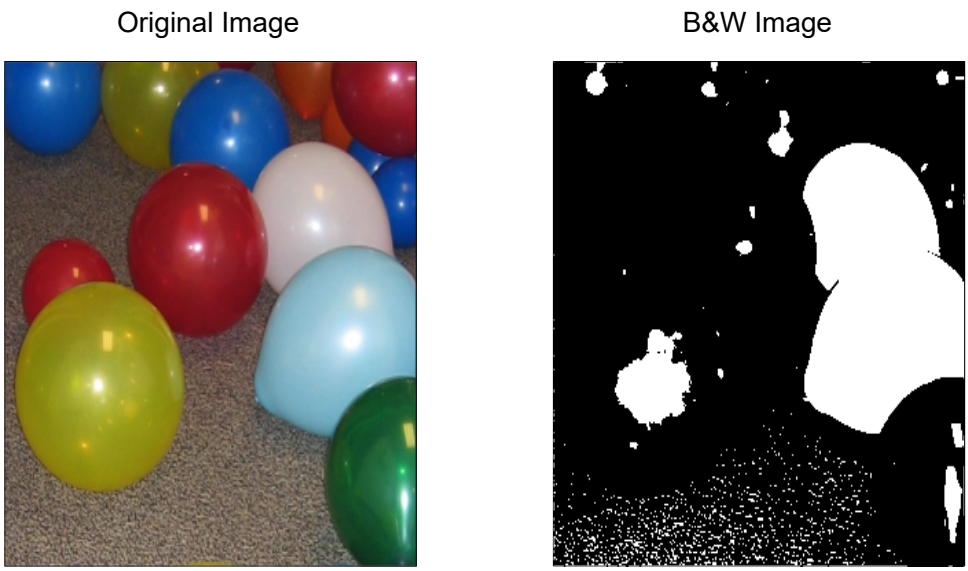
In [197]:

```
/* d. RGB to Grayscale Image */

gray_img = rgb2gray(img);

subplot(1,2,1), title('Original Image'), imshow(img);
subplot(1,2,2), title('Grayscale Image'), imshow(gray_img);
```



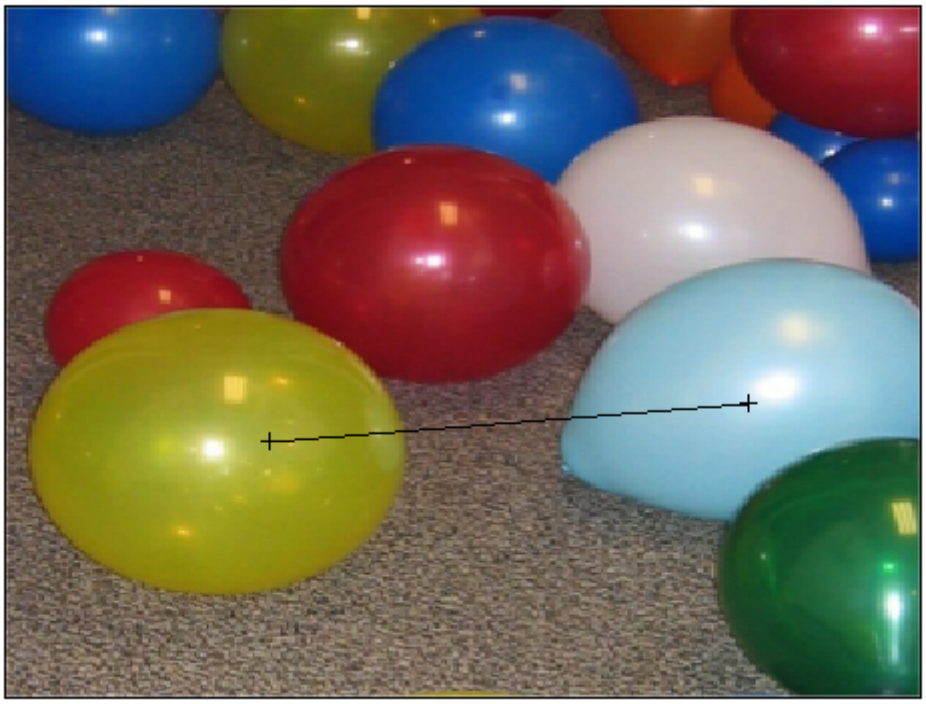
In [198]:

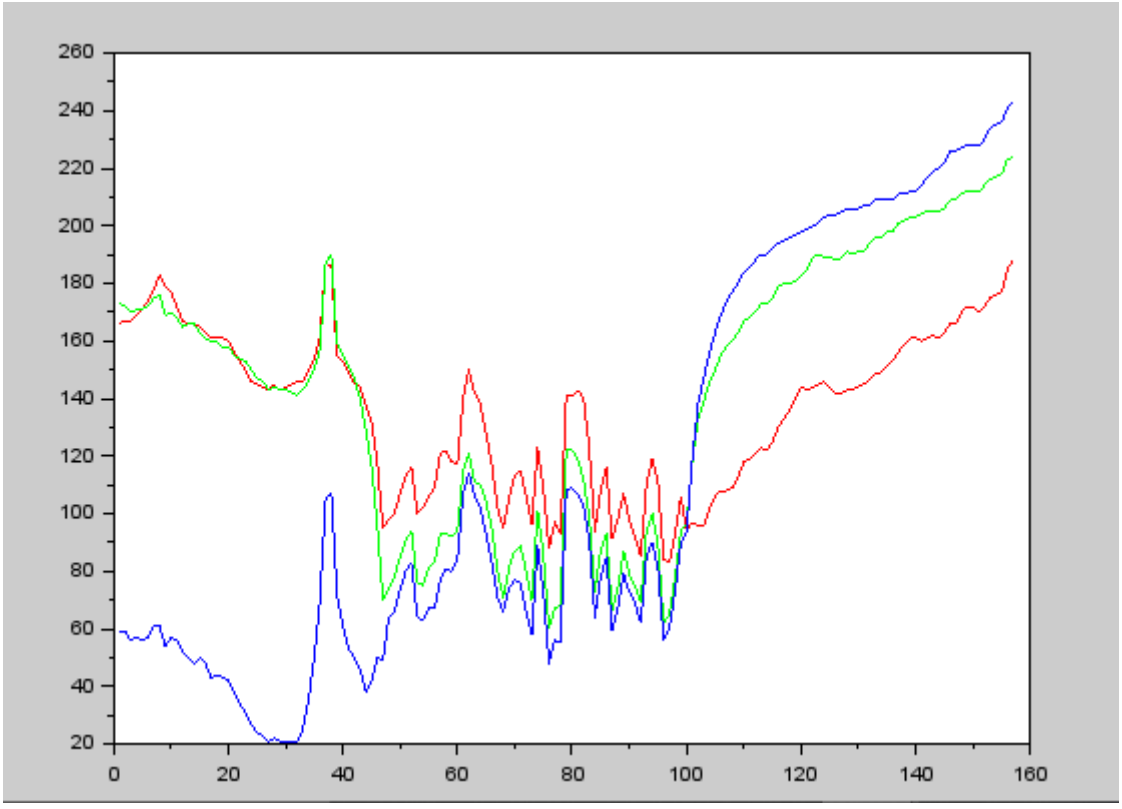
```
/* e. RGB to B&W Image */

bw_img = im2bw(img, 0.5);

subplot(1,2,1), title('Original Image'), imshow(img);
subplot(1,2,2), title('B&W Image'), imshow(bw_img);
```



In [50]:

```
/* f. Drawing Image Profile */
improfile(img);
```



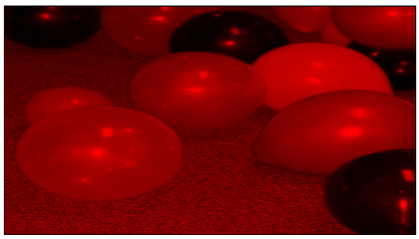


In [199]:

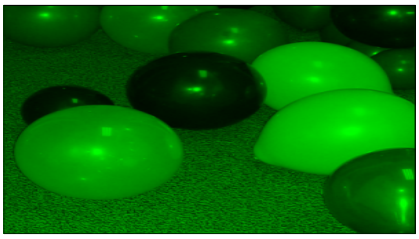
```
/* g. Seperating RGB Planes */

[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);
```

Original Image

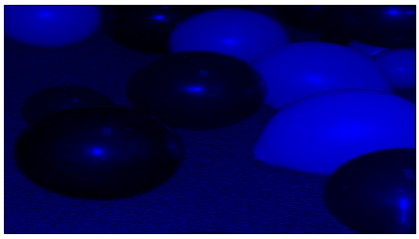
Red Plane Image



Green Plane Image

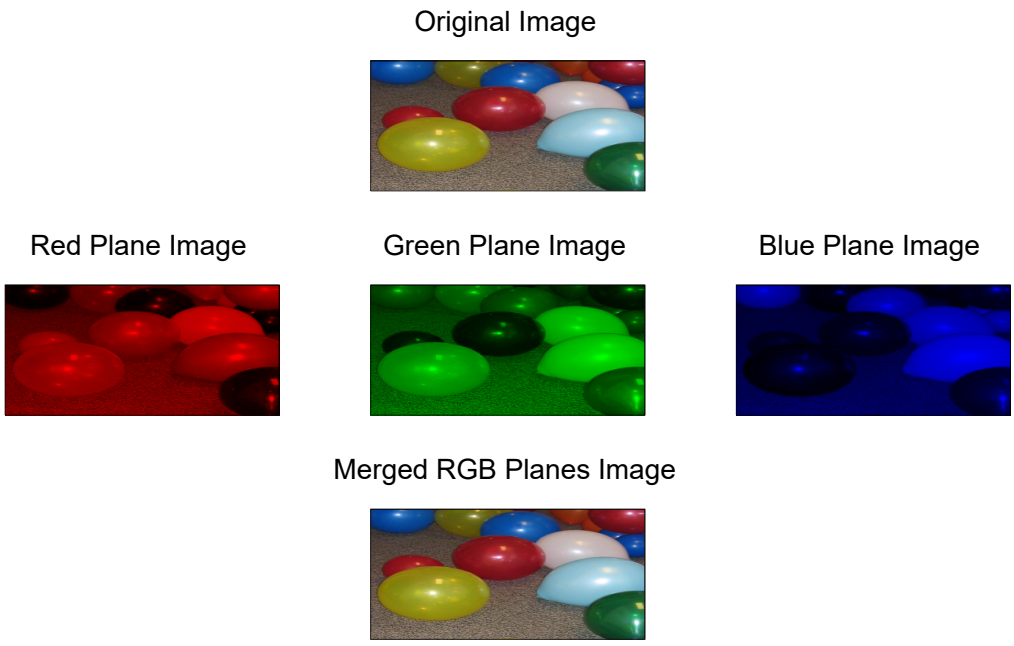


Blue Plane Image



In [200]:

```
/* h.  Creating image from RGB Planes */

// 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);
```



In [201]:

```
/* i. Flow control & loop in scilab */
for i=1:10
    if modulo(i,2) == 1 then
        disp(i);
    else
         continue;
    end,
end;
```

1.

3.

5.

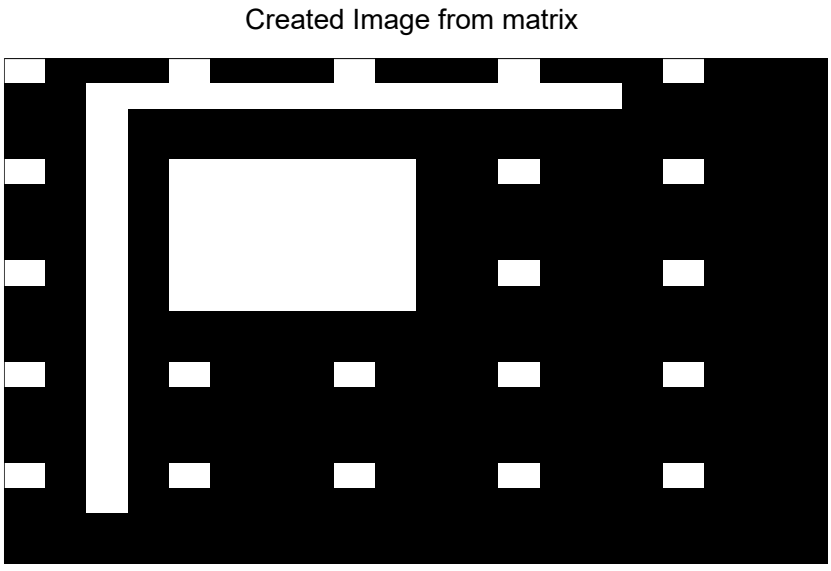
7.

9.

In [202]:

```
/* j. Create image with given 2D data */

mat = zeros(20,20,'uint8');
mat(5:10, 5:10) = 1;          // box
mat(2, 3:15) = 1;            //vertical line
mat(2:18, 3) = 1;           //horizontal line
mat(1:4:20,1:4:20) = 1;    // linear points

disp('Given 2D data : ', mat);
title('Created Image from matrix'), imshow(mat2gray(mat));
```

```
"Given 2D data : "

  1   0   0   0   1   0   0   0   1   0   0   0   1   0   0   0   1   0   0   0
  0   0   1   1   1   1   1   1   1   1   1   1   1   1   1   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  1   0   1   0   1   1   1   1   1   1   0   0   1   0   0   0   1   0   0   0
  0   0   1   0   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0
  1   0   1   0   1   1   1   1   1   1   0   0   1   0   0   0   1   0   0   0
  0   0   1   0   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  1   0   1   0   1   0   0   0   1   0   0   0   1   0   0   0   1   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  1   0   1   0   1   0   0   0   1   0   0   0   1   0   0   0   1   0   0   0
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```



Q. 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

Code)

In [189]:

```
/* a. Obtaining Negative Image */

img = imread('Test_images/download.jpeg');

subplot(1,2,1), title('Original Image'), imshow(img);

// for negative image use '~' operator or imcomplement() method
neg_img = 255 - img;
subplot(1,2,2), title('Negative Image'), imshow(neg_img);
```

Original Image

Negative Image

In [190]:

```
/* b. Obtaining flipped images */

subplot(2,2,1), title('Original Image'), imshow(img);
subplot(2,2,3), title('Up to Down Flipped Image'), imshow(flipdim(img, 1));
subplot(2,2,4), title('Left to right Flipped Image'), imshow(flipdim(img, 2));
```

Original Image

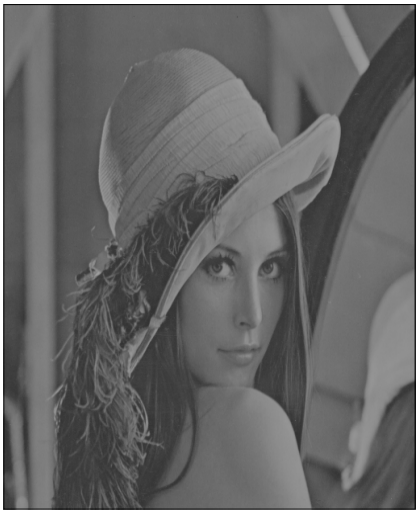
Up to Down Flipped Image

Left to right Flipped Image

In [191]:

```
/* c. Thresholding */

sam_img = imread('Test_images/Lena_dark.png');

subplot(1,2,1), title('Original Image'), imshow(sam_img);
subplot(1,2,2), title('Binary Image at D0 = 110'), imshow(im2bw(sam_img, 110/256));
```

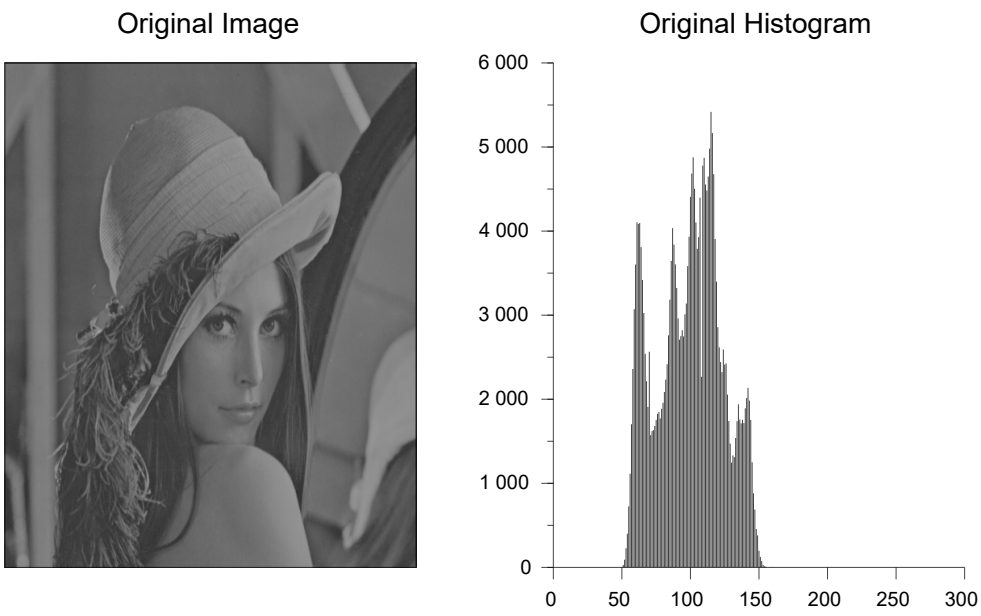
Original Image



Binary Image at D0 = 110

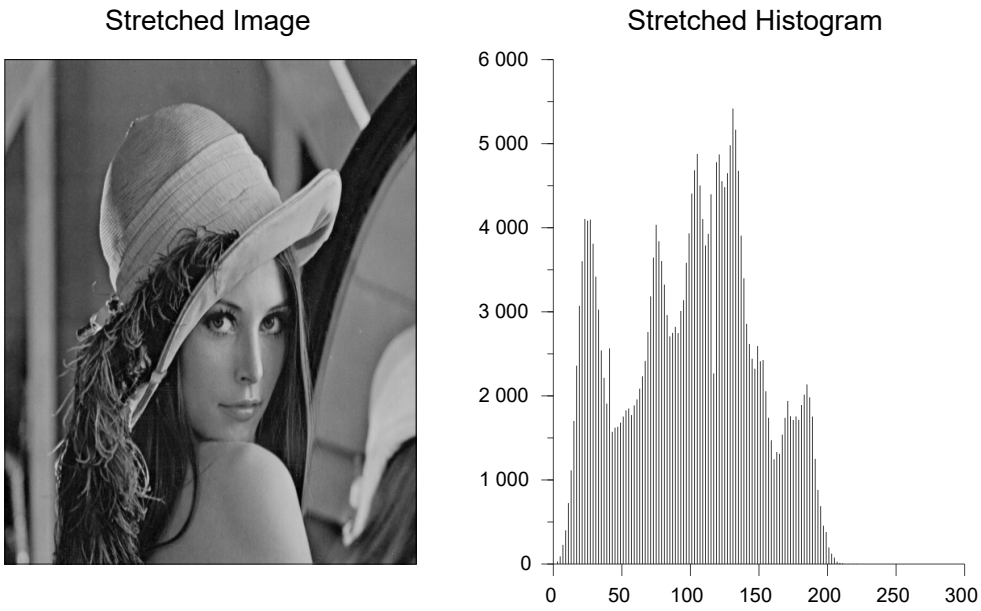
In [192]:

```
/* d. Contrast Stretching */
funcprot(0);

function image = contrast_stretch(gray_img)
a = min(gray_img(:));
b = max(gray_img(:));

image = (gray_img - a)*(255/(b-a));
endfunction;

c_sam = contrast_stretch(sam_img);

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



In [193]:

```
subplot(121),title("Stretched Image"),imshow(c_sam);
subplot(122),title("Stretched Histogram"), plot2d3(imhist(c_sam));
```



Q. 3) To write and execute programs for image arithmetic operations :

---

a. Addition of two images

b. Subtract one image from other image

c. Calculate mean value of image

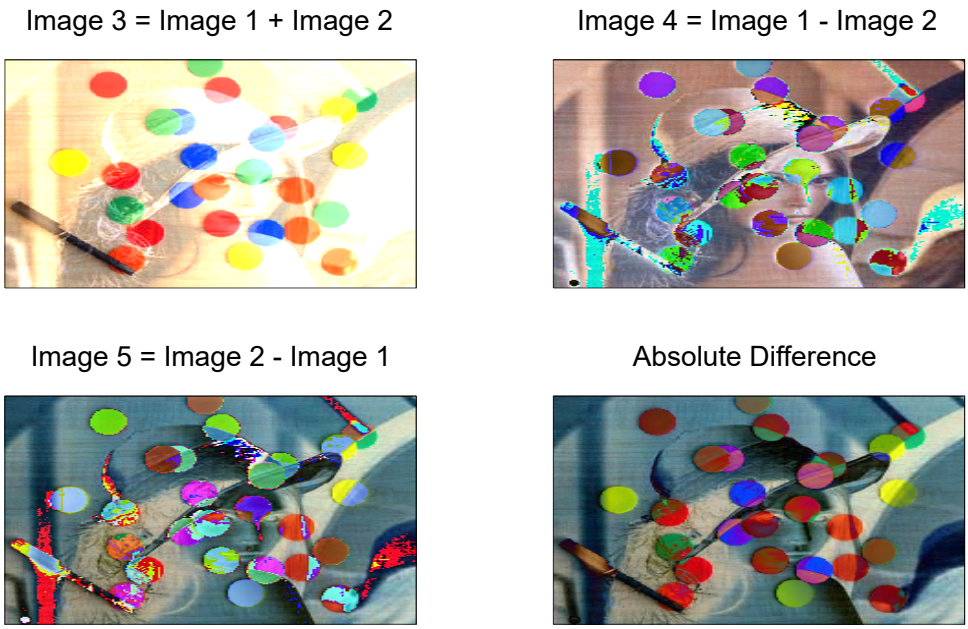
Code)

In [186]:

```
/* Sample images */

im1 = imread('Test_images/lena_1.jpeg');
im2 = imresize(imread('Test_images/coloredChips.png'), [225,225]);

subplot(1,2,1), title('Image 1'), imshow(im1);
subplot(1,2,2), title('Image 2'), imshow(im2);
```

In [187]:

```
// a. use '+' between images or use imadd() method
im3 = imadd(im1, im2);

// b. use '-' between images or use imsubtract() method

subplot(2,2,1), title('Image 3 = Image 1 + Image 2'), imshow(im3);
subplot(2,2,2), title('Image 4 = Image 1 - Image 2'), imshow(im1 - im2);
subplot(2,2,3), title('Image 5 = Image 2 - Image 1'), imshow(im2 - im1);
subplot(2,2,4), title('Absolute Difference'), imshow(imabsdiff(im1,im2));
```



In [188]:

```
/*c. Mean of the images */

printf('Mean of the image 1 : %.3f\n',mean(im2double(im1)));
printf('Mean of the image 2 : %.3f',mean(im2double(im2)));
```

```
Mean of the image 1 : 0.315

Mean of the image 2 : 0.666
```

Q. 4) To write and execute programs for image logical operations :

---

a. AND operation between two images

b. OR operation between two images

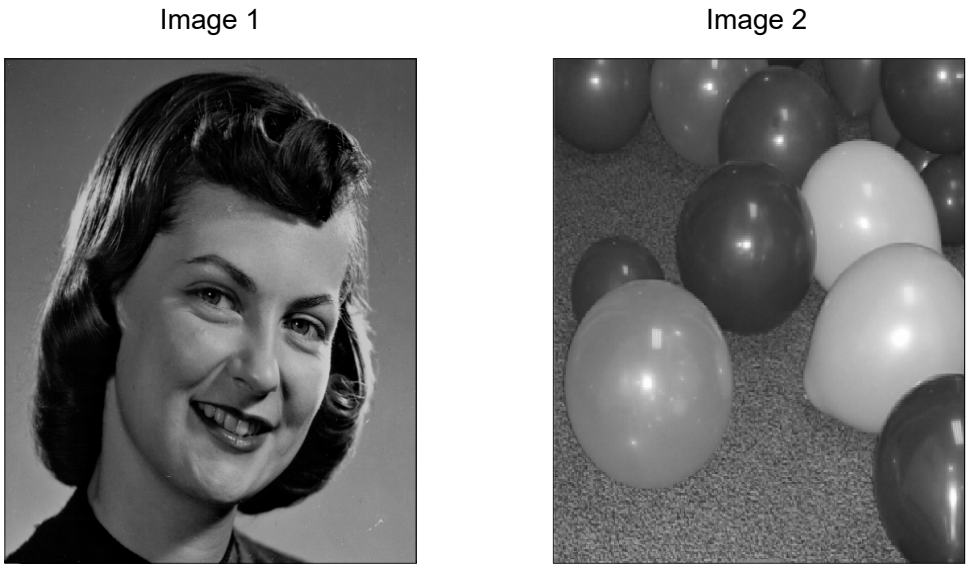
c. Calculate intersection of two images

d. NOT operation (Negative image)

Code)

In [178]:

```
/* Sample images */

// for airthmetic, images have to be of same size
img1 = imread('Test_images/girlface.bmp');
img2 = imread(fullpath(getIPCVpath() + '/images/balloons_gray.png'));
img2 = imresize(img2,[size(img1)]);

subplot(1,2,1), title('Image 1'), imshow(img1);
subplot(1,2,2), title('Image 2'), imshow(img2);
```



In [179]:

```
/* a. And operation on two images */
subplot(1,2,1), title('Image 1 & Image 2'), imshow(bitand(img1, img2));

/* b. Or operation on two images */
subplot(1,2,2), title('Image 1 | Image 2'), imshow(bitor(img1,img2));
```

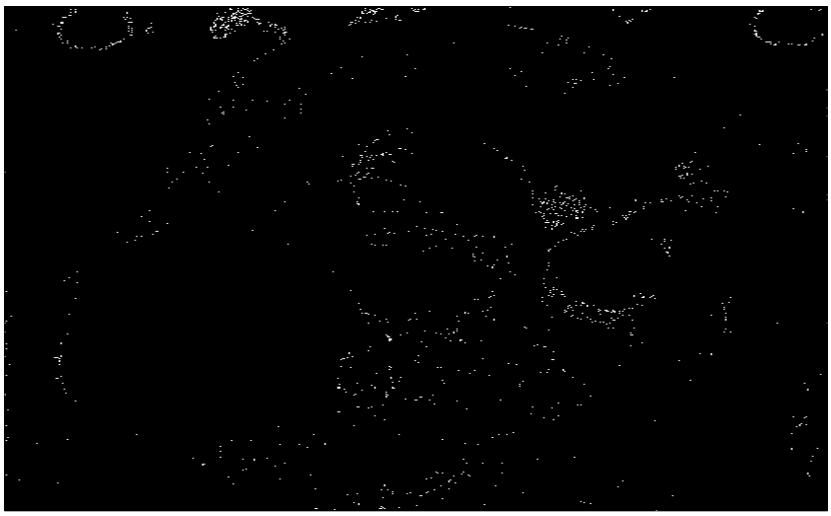
Image 1 & Image 2

Image 1 | Image 2

In [180]:

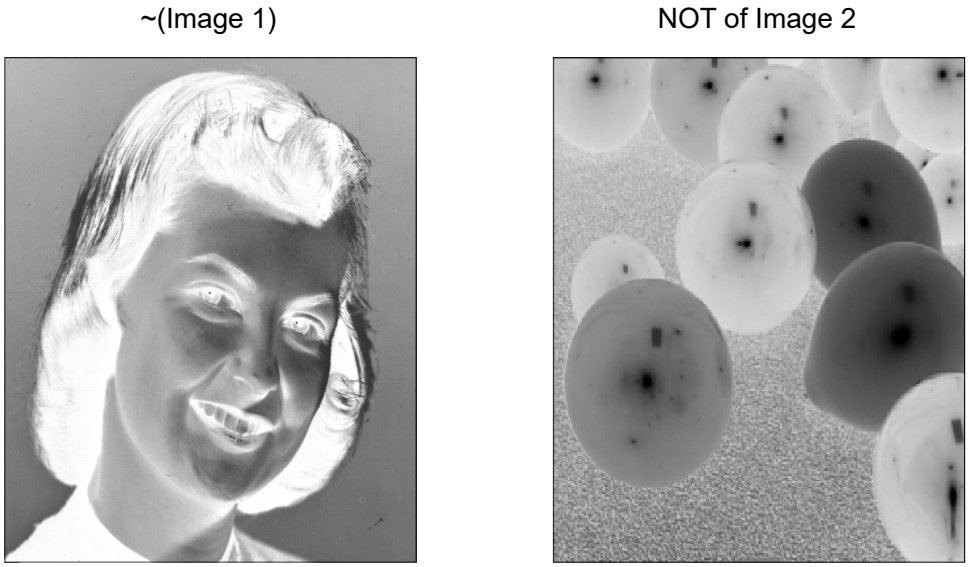
```
/* c. Intersection of two images */

inter_img = (img1 - img2) == 0;
title('Intersection of Image 1 & Image 2'), imshow(inter_img);
```

Intersection of Image 1 & Image 2



In [181]:

```
/* d. NOT on the images */

subplot(1,2,1), title('~(Image 1)'), imshow(~img1);
subplot(1,2,2), title('NOT of Image 2'), imshow(~img2);
```



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

---

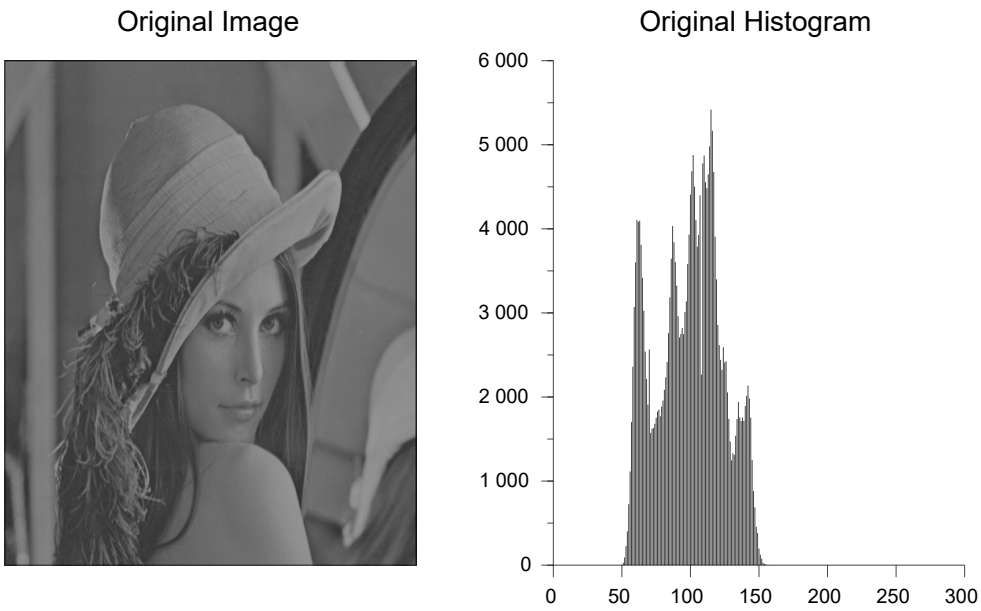
a. Standard MATLAB function

b. Program without using standard MATLAB functions

Code)

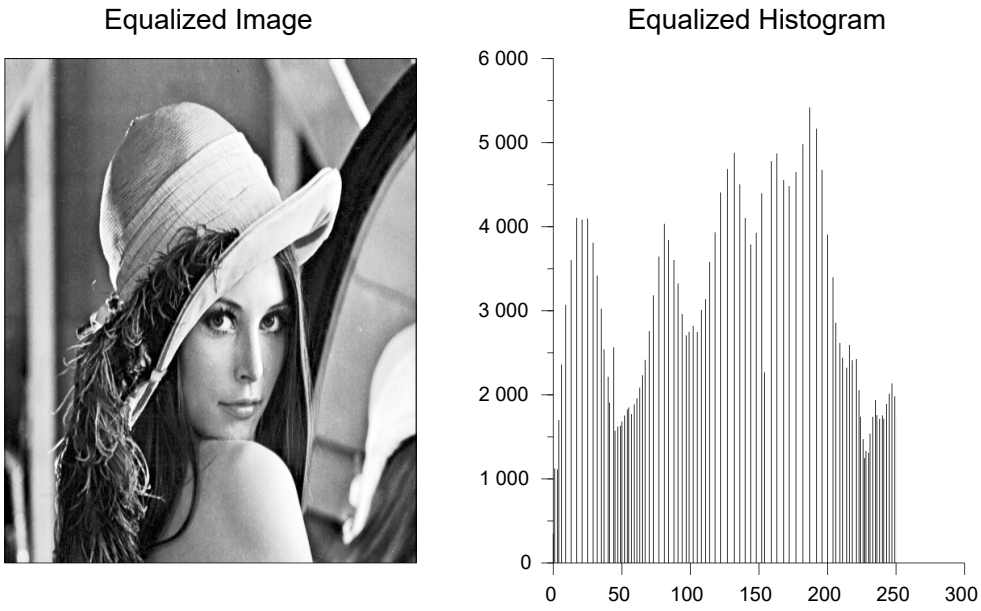
In [182]:

```
/* 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);
```



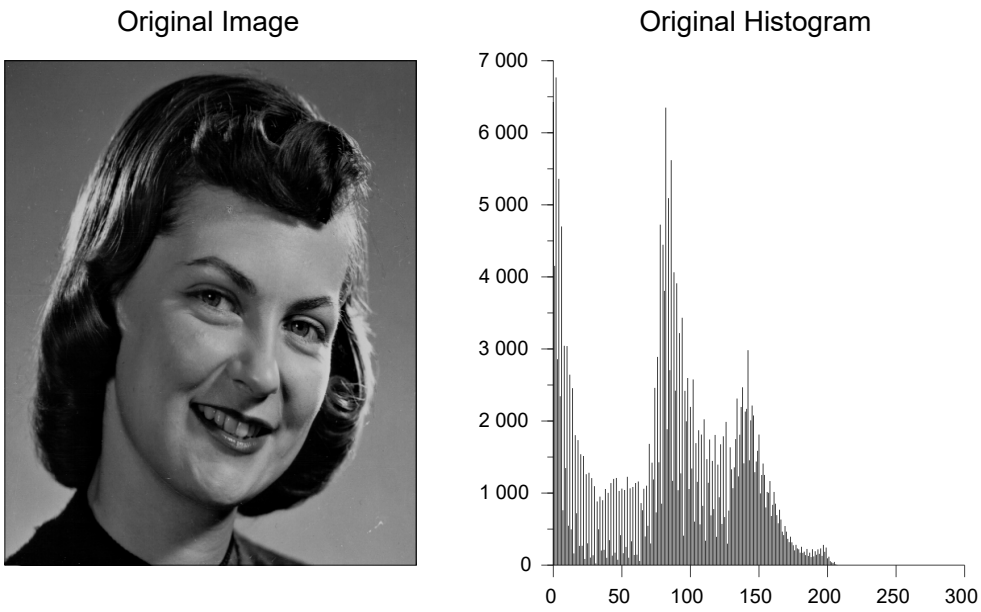
In [183]:

```
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);
```



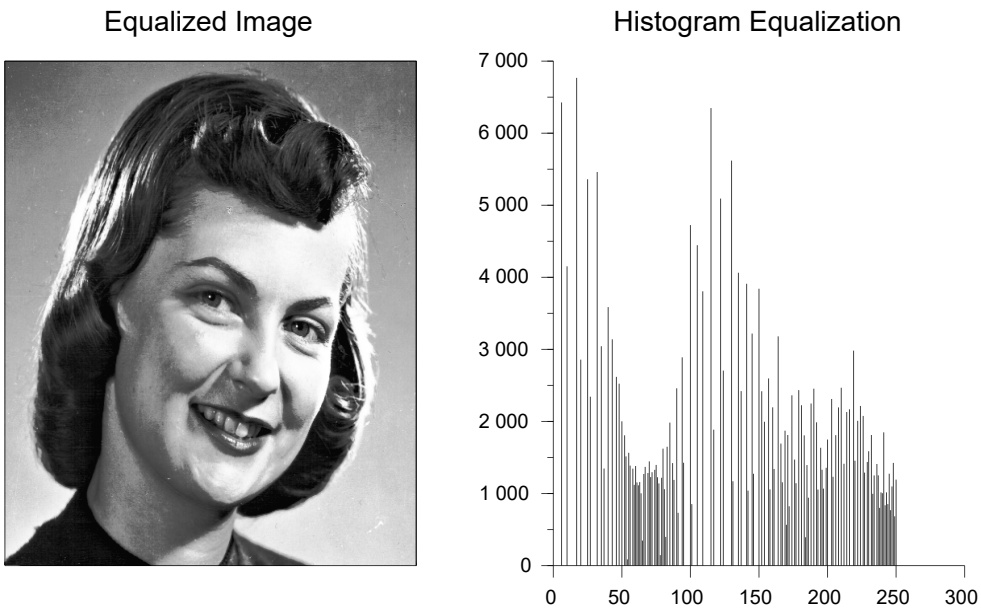
In [184]:

```
/* 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);
```



In [185]:

```
/* 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);
```



Q. 6) To write and execute program for geometric transformation of image :

a. Translation

b. Scaling

c. Rotation

d. Shrinking

e. Zooming

Code)

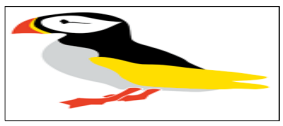
In [6]:

```
/* a. Translation */

S1 = imread('Test_images/lena.jpeg');

// Translation for x  = 20

mat = [  1 0 0;...
         0 1 0;...
        20 0 1];
S2 = imtransform(S1,mat,'affine');

// Translation for y  = -20

mat(3, 1:2) = [0 -20];
S3 = imtransform(S1,mat,'affine');

// 'Translation for (-20, 30)'

mat(3, 1:2) = [-20 30];
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);
```

In [7]:

```
/* b. Scaling */

s_img = imread(fullpath(getIPCVpath() + "/images/puffin.png"));

width = size(s_img, 'c');      // column pixels = width
height = size(s_img, 'r');    // row pixels = height

// Scaling width by 2

mat = [ 2 0;
        0 1;
        0 0];

sc1 = imtransform(s_img, mat, 'affine', width*mat(1), height*mat(5));

// Scaling height by 2

mat([1,5]) = [1 2];

sc2 = imtransform(s_img, mat, 'affine', width*mat(1), height*mat(2));

// Scaling image by 2

mat([1,5]) = [2 2];

sc3 = imtransform(s_img, mat, 'affine', width*mat(1), height*mat(2));

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

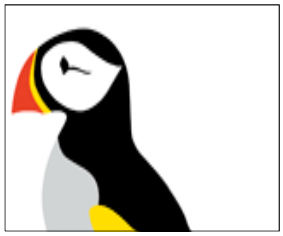


Size : 192 * 161 * 3

Image scaling width by 2

Size : 192 * 322 * 3

Image scaling height by 2



Size : 192 * 161 * 3

Image scaling by 2

Size : 192 * 161 * 3

In [8]:

```
/* c. Rotation */

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));
```

Original Image

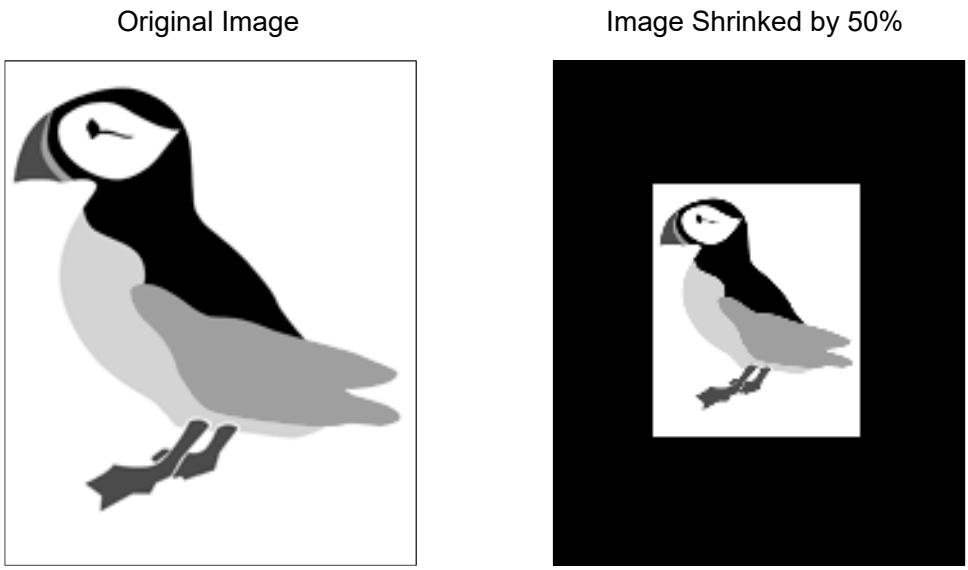
Image rotation by 45

Image rotaion by -45

Image rotaion by 180

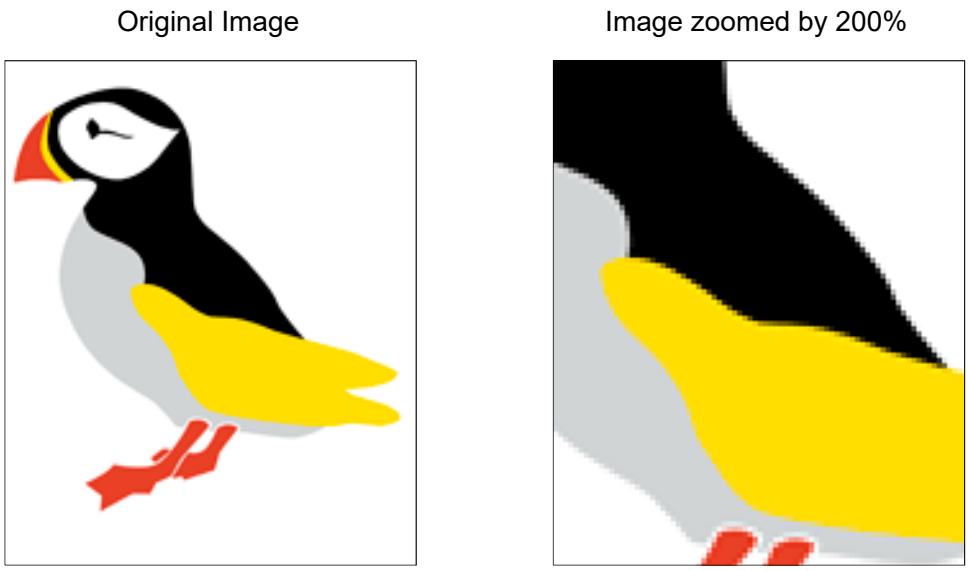
In [55]:

```
/* Shrinking */

[r c] = size(s_img);
f = 0.5;
im_50 = zeros(r, c, 'uint8');
shrinked = rgb2gray(imresize(s_img, f));
im_50(48:143, 40:120) = shrinked;


subplot(121), title('Original Image'), imshow(rgb2gray(s_img));
subplot(122), title('Image Shrinked by 50%'), imshow(im_50);
```



In [59]:

```
/* Zooming */

f = 2;
im2 = imresize(s_img, f);
subplot(121), title('Original Image'), imshow(s_img);
subplot(122), title('Image zoomed by 200%'), imshow(im2(96:287, 81:241, :));
```



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

---

a. image restoration

b. Remove Salt and Pepper Noise

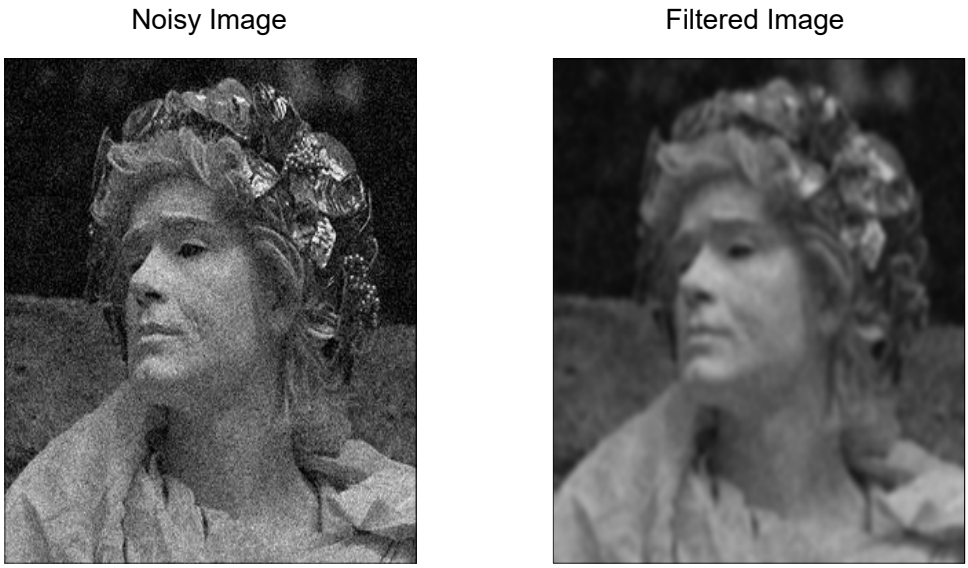
c. Minimize Gaussian noise

d. Median filter

Code)

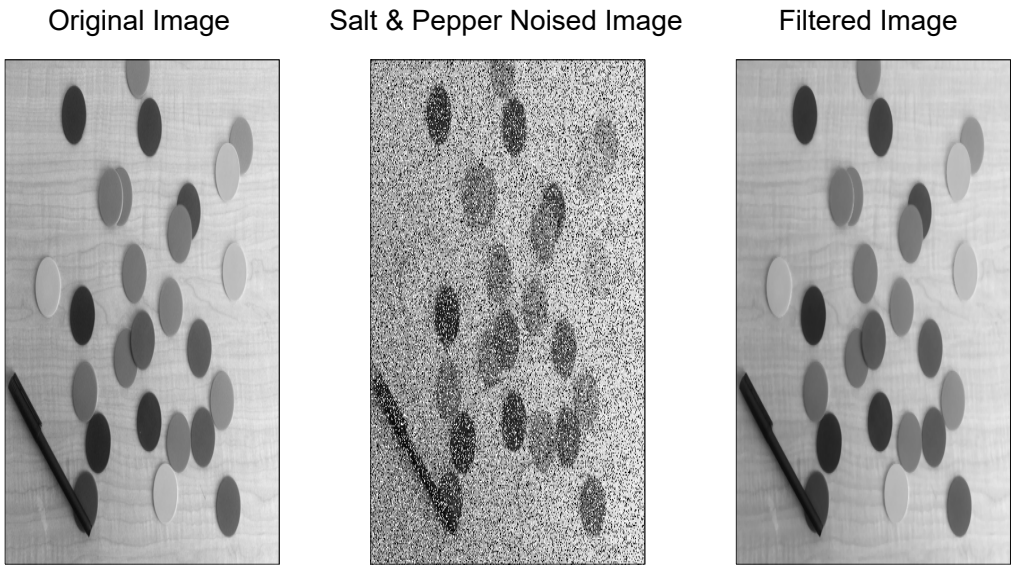
In [15]:

```
/* 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));
```



In [47]:

```
/* 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));
```

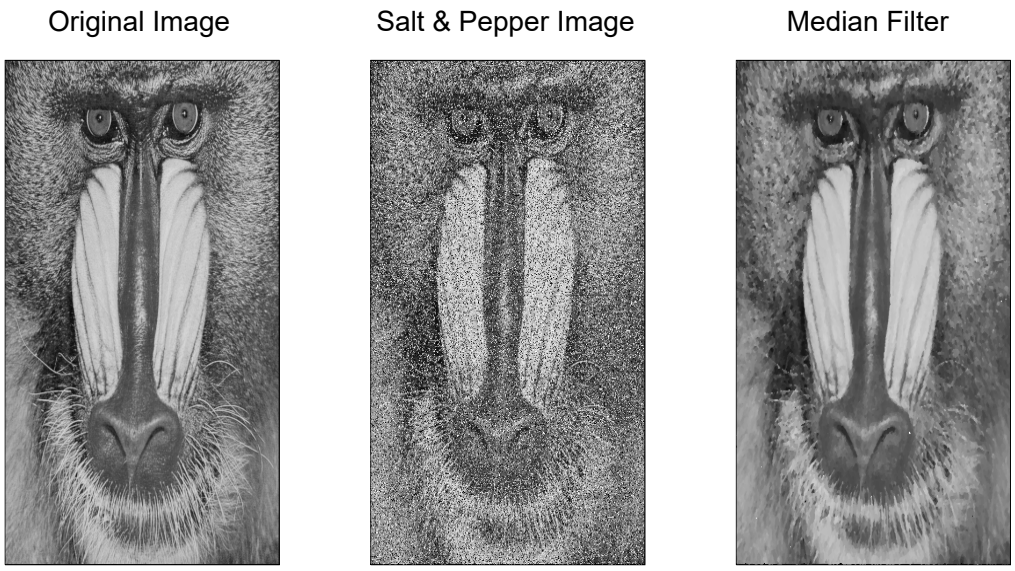


In [46]:

```
/* 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));
```

In [44]:

```
/* 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));
```



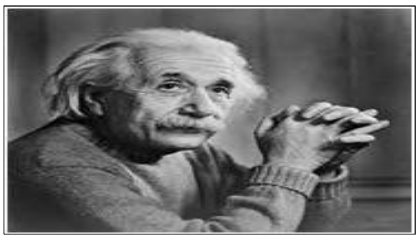
Q. 8) Write and execute programs to use spatial low pass and high pass filters.

Code)

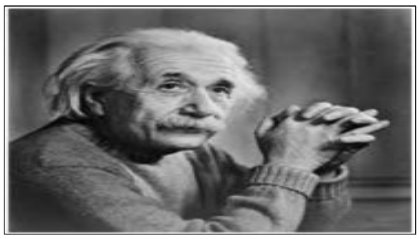
In [60]:

```
/* Spatial Low Pass Filter */

i1 = imread('Test_images/einstein.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);
```

Original Image

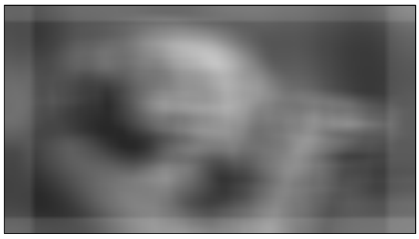


Default Gaussian kernel



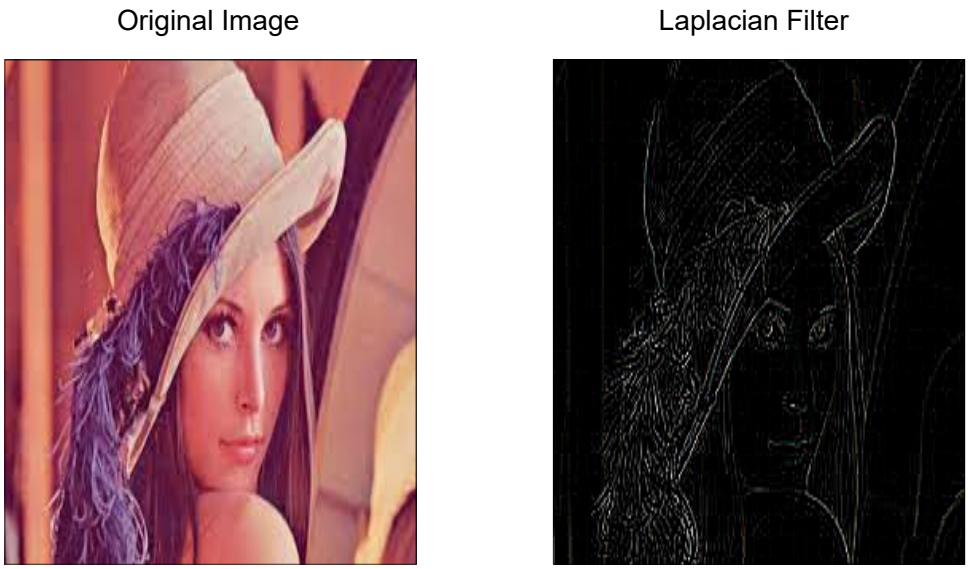
Gaussian kernel with 8 * 8 with sigma = 10

Gaussian kernel with 25 * 25 with sigma = 31



In [61]:

```
/* Spatial Low Pass Filter */

i1 = imread('Test_images/lena.jpeg');

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

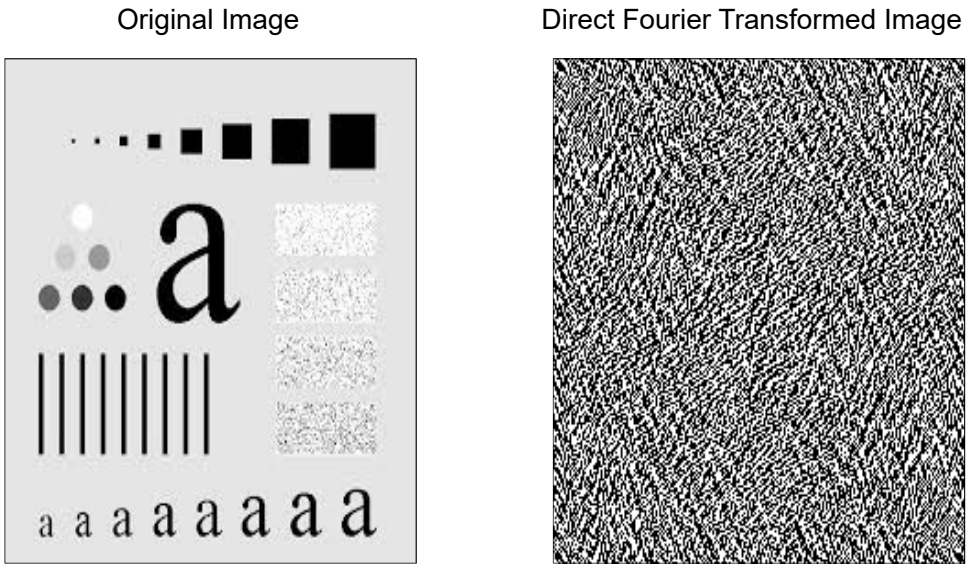
b. Perform low pass and high pass filtering in frequency domain

c. Apply IFFT to reconstruct image

Code)

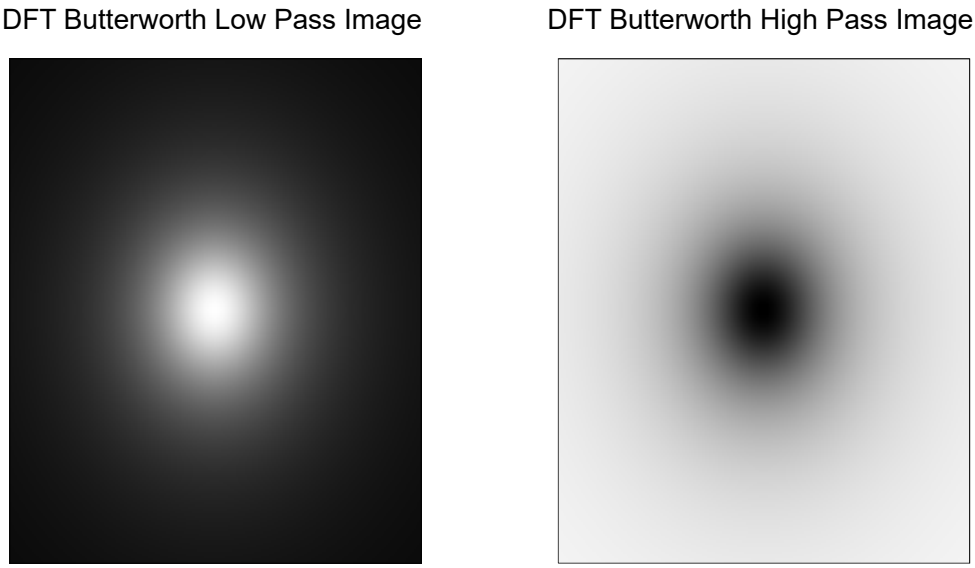
In [117]:

```
/* a. FFT on image */

img = rgb2gray(imread('Test_images/sample.jpeg'));
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);
```



In [118]:

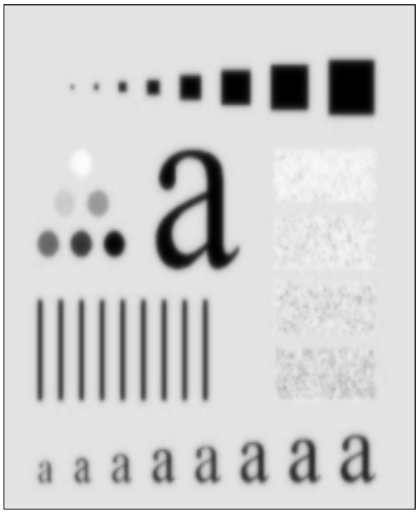
```
/* b. low & high pass filtering */
// Butterworth Filters
G11 = mkfftfilter(img, 'butterworth1', 0.3);
H11 = 1 - G11;

subplot(121), title('DFT Butterworth Low Pass Image'), imshow(G11);
subplot(122), title('DFT Butterworth High Pass Image'),imshow(H11);
```



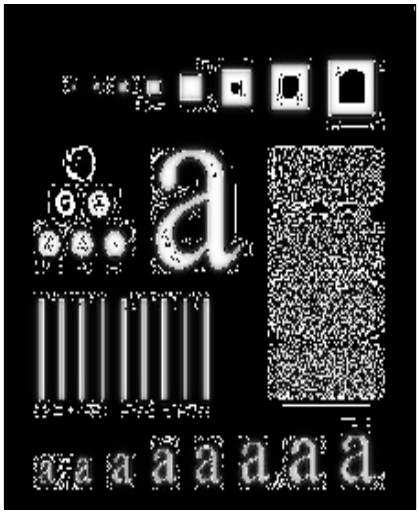
In [119]:

```
S2 = ft_img .* fftshift(G11);
bwh_l = uint8(ifft(S2));
S2 = ft_img .* fftshift(H11);
bwh_h = uint8(ifft(S2));

subplot(121), title('DFT Butterworth Low Pass Image'), imshow(bwh_l);
subplot(122), title('DFT Butterworth High Pass Image'),imshow(bwh_h);
```

DFT Butterworth Low Pass Image



DFT Butterworth High Pass Image



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

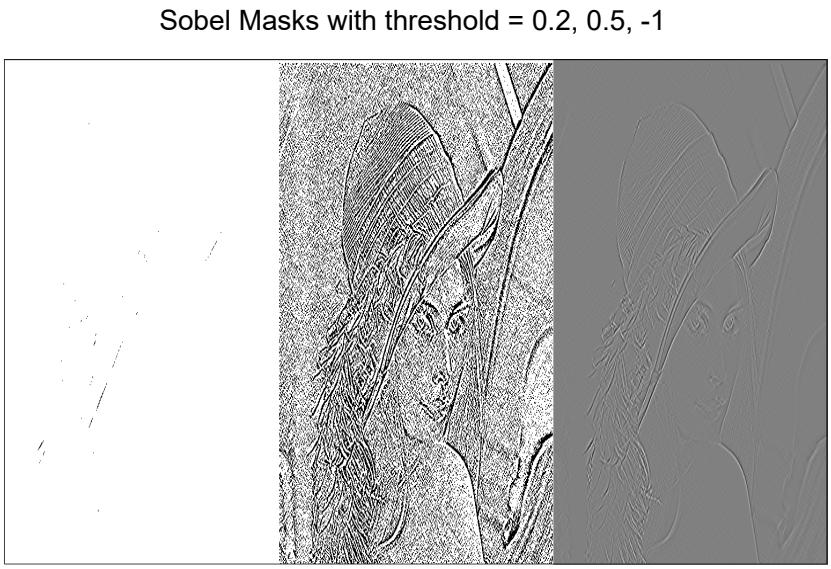
Code)

In [151]:

```
// Edge Detection Masks

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

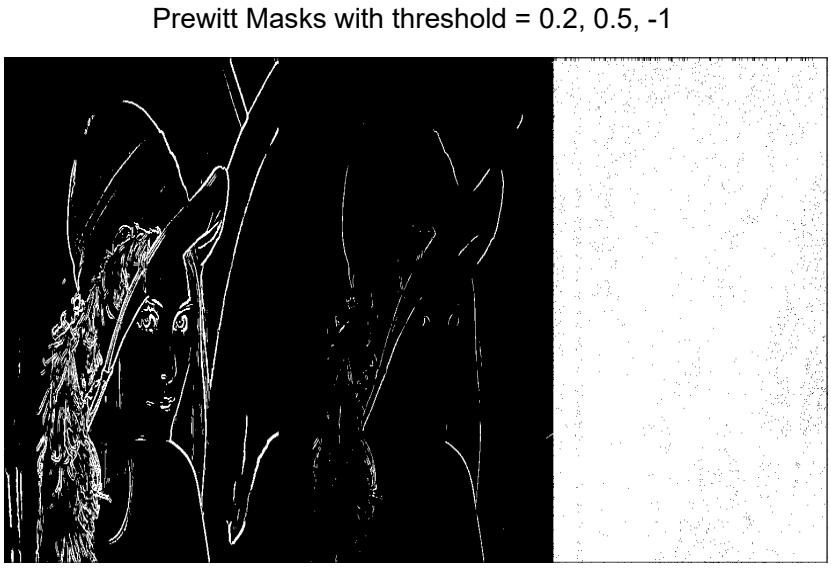
In [170]:

```
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]);
```



In [173]:

```
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]);
```

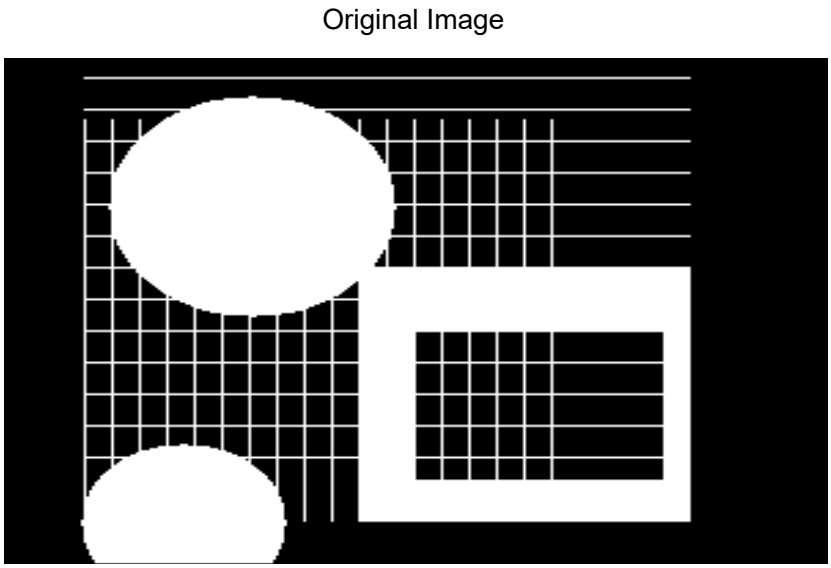


Q. 11) Write and execute program for image morphological operations erosion and dilation.

Code)

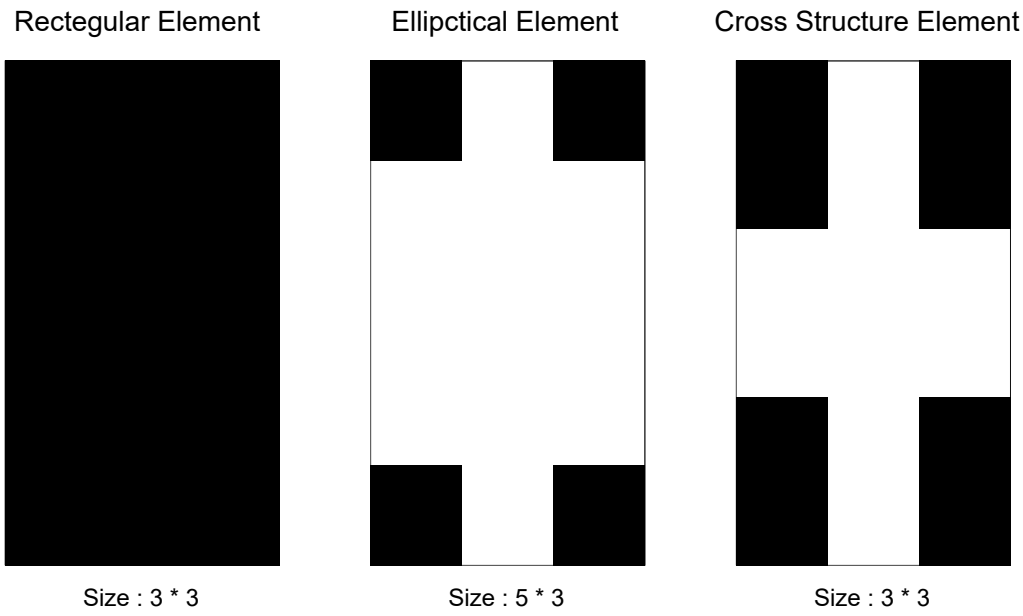
In [51]:

```
// Morphological Operation : Erosion & Dilation

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

// Image
c = im2bw(imread('Test_images/circlesBrightDark.png'), 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);
```



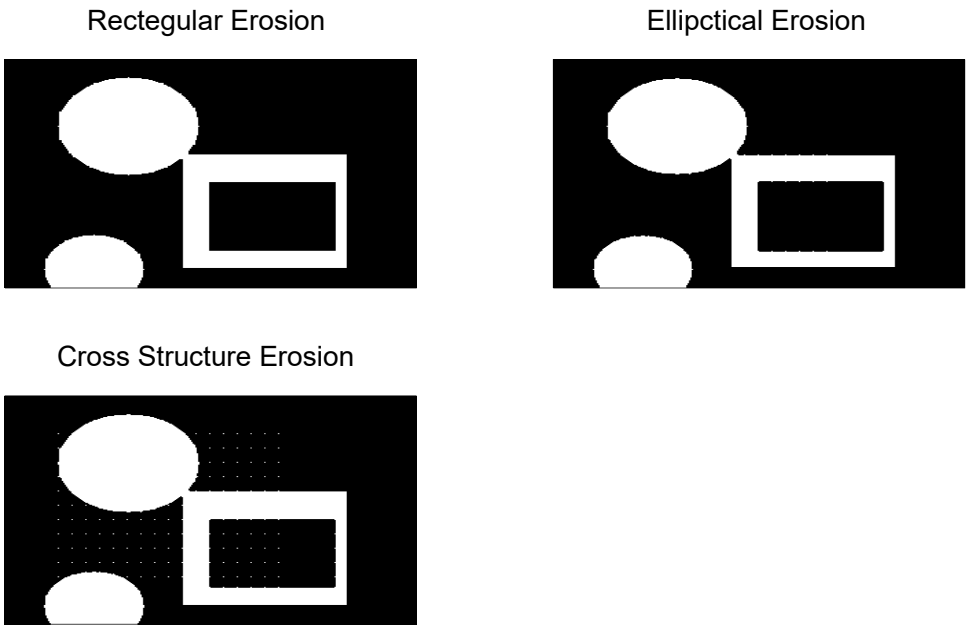
In [56]:

```
// Structure element
s1 = imcreatese('rect', 3, 3);
s2 = imcreatese('ellipse', 5, 3);
s3 = imcreatese('cross', 3, 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));
```



In [55]:

```
// erosion
e1 = imerode(c, s1);
e2 = imerode(c, s2);
e3 = imerode(c, s3);

// Plotting
subplot(2,2,1), title('Rectegular Erosion'), imshow(e1);
subplot(2,2,2), title('Ellipctical Erosion'), imshow(e2);
subplot(2,2,3), title('Cross Structure Erosion'), imshow(e3);
```



In [57]:

```
// dilation
d1 = imdilate(c, s1);
d2 = imdilate(c, s2);
d3 = imdilate(c, s3);

// Plotting
subplot(2,2,1), title('Rectegular Dilation'), imshow(d1);
subplot(2,2,2), title('Ellipctical Dilation'), imshow(d2);
subplot(2,2,3), title('Cross Structure Dilation'), imshow(d3);
```