Practical 1

Aim- perform correlation of two matrices. Take user input for two matrices size and the values. Then show the output of correlation of matrices.

Code-

```
clc;
disp('CORRELATION');
disp('Accept 2 2d signals in the form matrices from the user and perform correlation between them');
disp('My first Matrix')
disp('Enter dimension of the matrix')
m=input('enter row value')
n=input('enter column value')
for i =1:m
   for j=1:n
      x(i,j)=input('enter first matrix value ')
   end
end
for i =1:m
   for j=1:n
      t(i,j)=input('enter second matrix value ')
   end
end
h=t($:-1:1,:)
h3=h(:,$:-1:1)
y=conv2(x,h3)
disp('Correlation of matrix x and t is')
disp(y)
```

input-

matrix rows 3

matrix columns 3

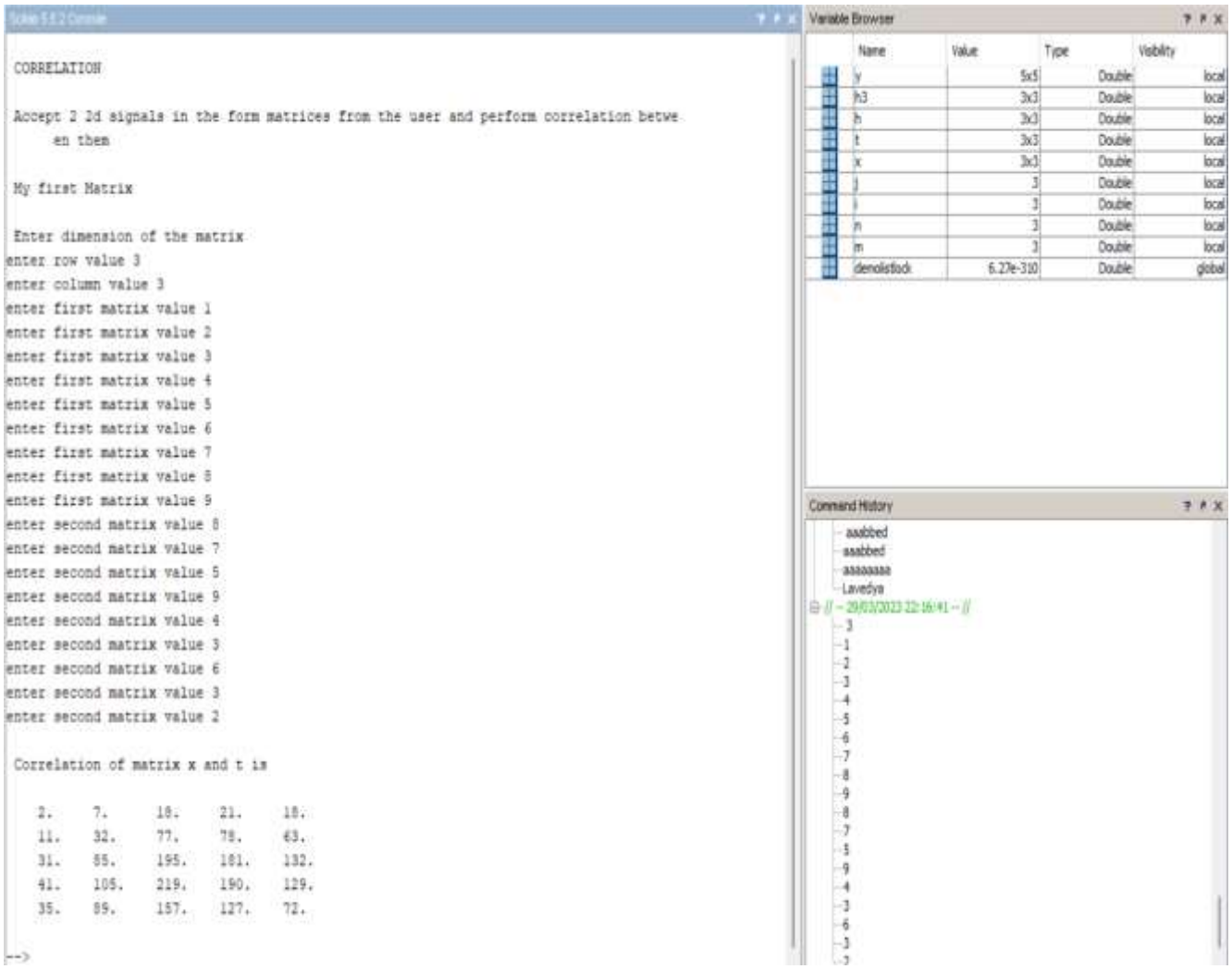
Matrix 1 values

1	2	3
4	5	6
7	8	9

Matrix 2 values

8	7	5
9	4	3
6	3	2

Output-



Practical 2

Aim- The aim of this code is to perform 2D linear convolution between two matrices. Ask user to input the dimensions of the first matrix and its values, followed by the dimensions and values of the second matrix. Then perform linear convolution between the two matrices using the conv2 function in SCILAB and displays the resulting matrix.

Code-

```
clc;
disp('2D LINEAR CONVOLUTION');
disp('Accept 2 2d matrices from the user and perform linear convolution between them');
disp('My first Matrix')
disp('Enter dimension of the matrix')
m=input('enter row value')
n=input('enter column value')
for i =1:m
   for j=1:n
      x(i,j)=input('enter first matrix value ')
   end
end
disp('Matrix X is =');
disp(x)

for i =1:m
   for j=1:n
      t(i,j)=input('enter second matrix value ')
   end
end
disp('Matrix T is =');
disp(t)
A=conv2(x,t)
disp('convolution of matrix x and t is')
disp(A)
```

input-

matrix rows 2

matrix columns 2

Matrix 1 values

4	7
5	2

Matrix 2 values

8	4
3	2

Output-

```
 2D LINEAR CONVOLUTION

 Accept 2 2d matrices from the user and perform linear convolution between them

 My first Matrix

 Enter dimension of the matrix
enter row value 2
enter column value 2
enter first matrix value 4
enter first matrix value 7
enter first matrix value 5
enter first matrix value 2

 Matrix X is =

    4.    7.
    5.    2.
enter second matrix value 8
enter second matrix value 4
enter second matrix value 3
enter second matrix value 2

 Matrix T is =

    8.    4.
    3.    2.

 convolution of matrix x and t is

    32.    72.    28.
    52.    65.    22.
    15.    16.    4.

-->
```

Practical 3

Aim- Accept two 2D matrices from the user, performs 2D Circular convolution between them using the conv2 function, and then performs 2D circular convolution on the result. The aim of the code is to demonstrate the implementation of 2D circular convolution using SCILAB.

Code-

```
clc;
disp('2D CIRCULAR CONVOLUTION');
disp('Accept 2 2d signals in the form matrices from the user and perform circular
convolution between them');
disp('My first Matrix')
disp('Enter dimension of the matrix')
m=input('enter row value')
n=input('enter column value')
for i =1:m
   for j=1:n
      x(i,j)=input('enter first matrix value ')
   end
end
disp('Matrix X is =');
disp(x)

for i =1:m
   for j=1:n
      t(i,j)=input('enter second matrix value ')
   end
end
disp('Matrix T is =')
disp(t)
A=conv2(x,t)
y1=[A(:,1) + A(:,$),A(:,2)];
y2=[y1(1,:)+y1($,:);y1(2,:)];
disp('convolution of matrix x and t is')
disp(A)
disp(y2, ' result of circular')
imwrite('C:\Users\admin\Desktop\dip\abc.jfif')
```

Input-

matrix rows 2

matrix columns 2

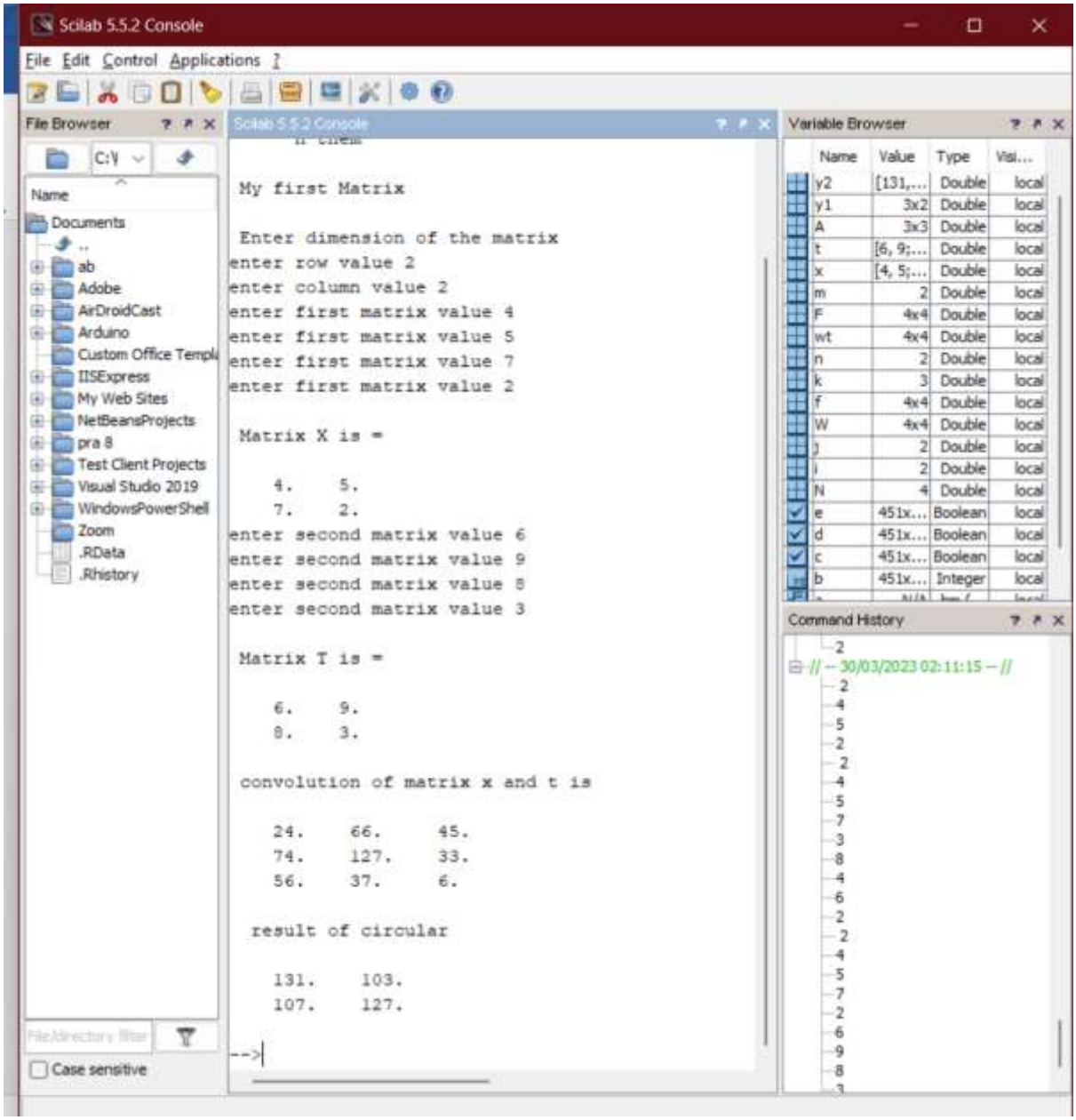
Matrix 1 values

4	5
7	2

Matrix 2 values

6	9
8	3

Output-



Practical 4

Aim- read an image file, display it, create a negative version of the image, and display the negative image

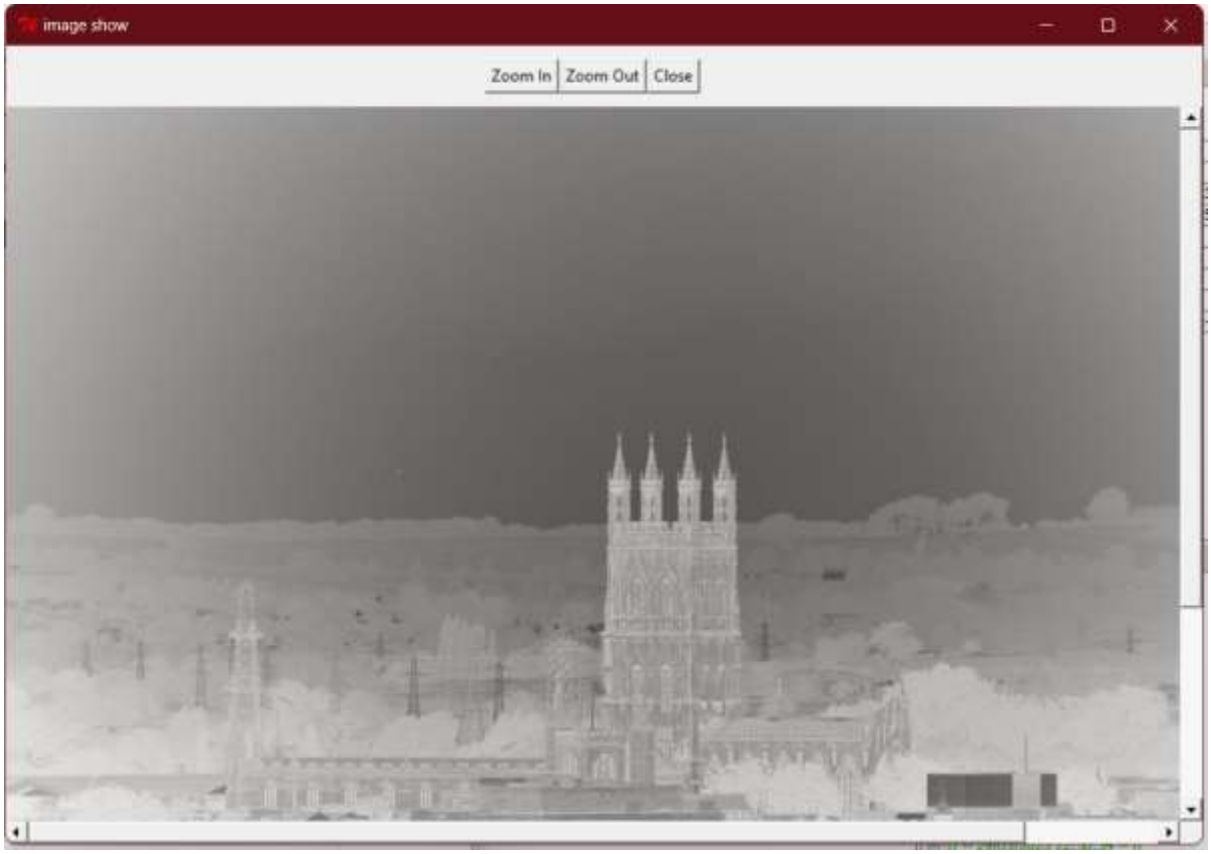
Code-

```
xdel(winsid())
a=imread("C:\Users\admin\Desktop\dip\16cdf5e1-c820-48b2-b8ae-44ccdad203e7.jfif")
imshow(a)
figure(2)
k=255-a
k=uint8(k)
imshow(k)
title("Negative")
```

Output-

Original image-

Negative image-



Practical 5

Aim-Brighten the image by adding 50 to each pixel value of the image to increase the brightness, converts it to an unsigned 8-bit integer format, displays the new image and saves the image in JPEG file format to the specified location.

Code-

```
xdel(winsid());
a=imread("C:\Users\admin\Desktop\dip\16cdf5e1-c820-48b2-b8ae-44ccdad203e7.jfif")
b=a+50
b=uint8(b)
imshow(b)
imwrite(b,'C:/Users/admin/Desktop/dip/as.jfif');
```

Output-

Original Image-

Modified Image=

.

Practical 6

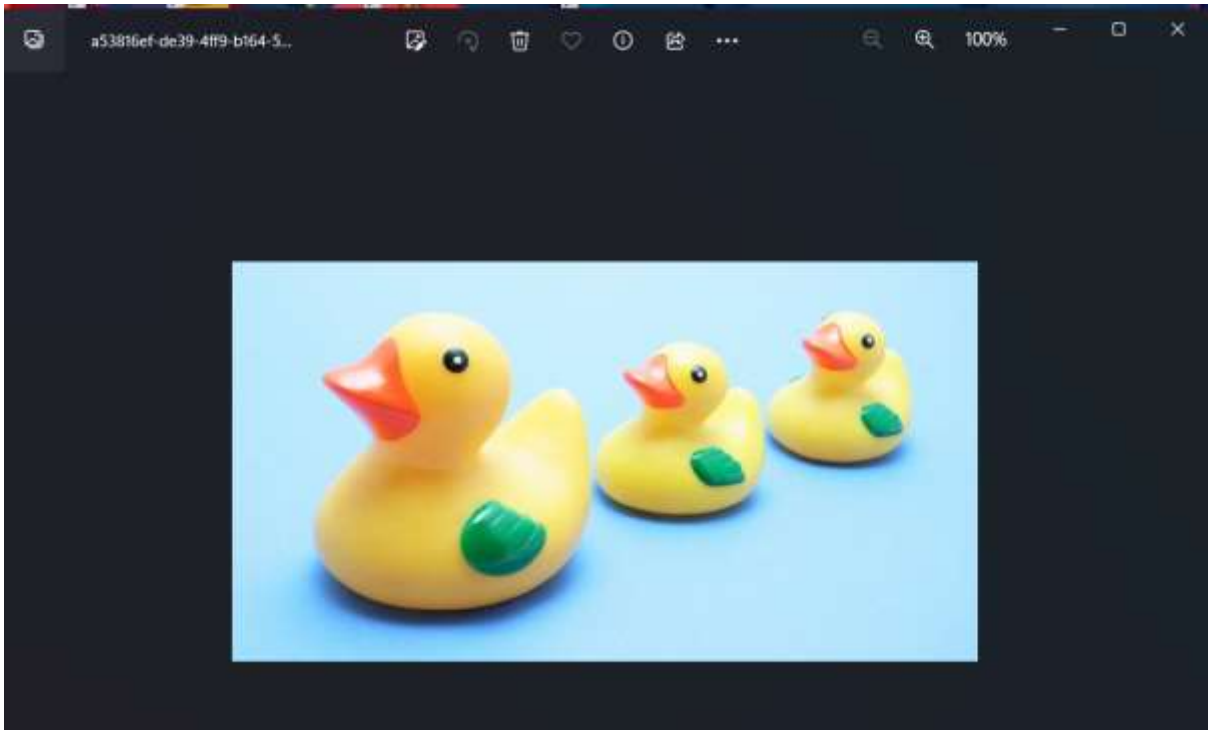
Aim- demonstrate how to increase and decrease the contrast of an image in SCILAB.

Code-

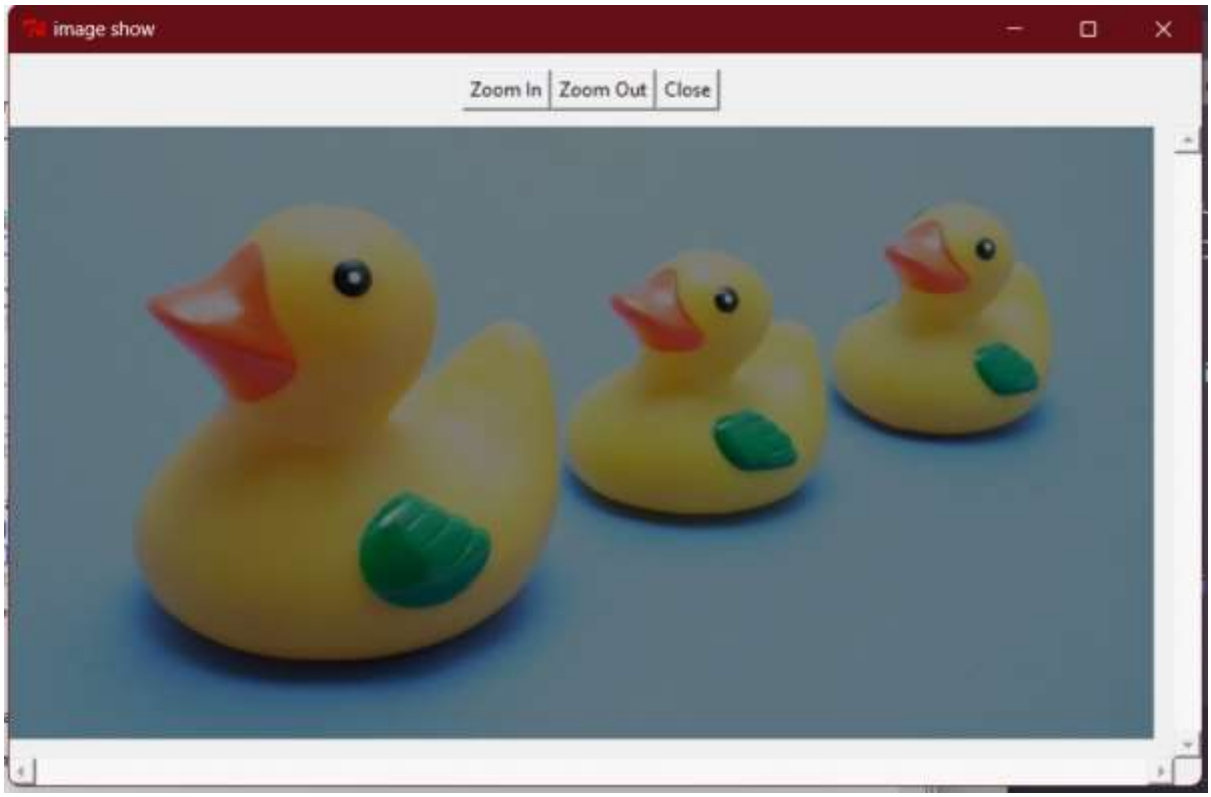
```
xdel(winsid())
a=imread("C:\Users\admin\Desktop\dip\a53816ef-de39-4ff9-b164-5e9105ceccb3.jfif")
figure(1)
imshow(a)
title("Original image")
figure(2)
b=double(a)*0.5
b=int8(b)
imshow(b)
title("Contrast increased")
figure(3)
c=double(a)*0.5
c=uint8(c)
imshow(c)
title("Contrast decreased")
imwrite(c,'C:\Users\admin\Desktop\dip\ouput.jpg',jpg)
```

Ouput-

Original Image-



Modified Image-



Practical 7

Aim- perform thresholding on a grayscale image and save It on desktop using “imwrite” command.

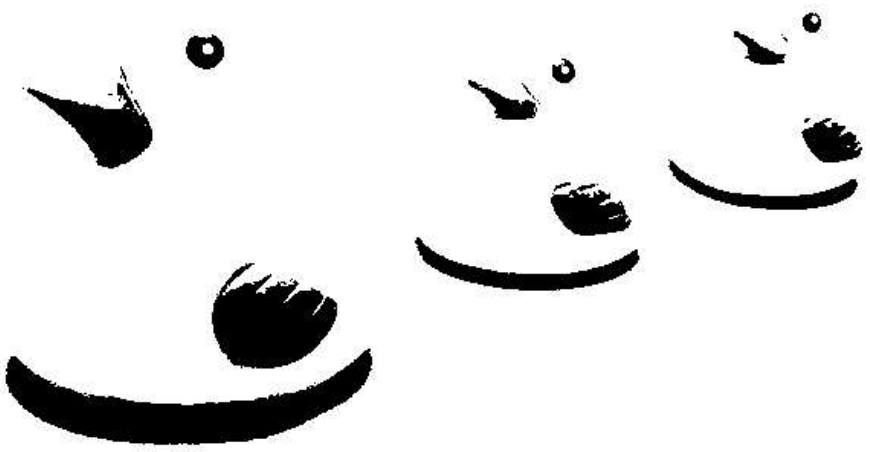
Code-

```
clc;
a = imread("C:\Users\admin\Desktop\output.jpg");
a = rgb2gray(a);
[m, n] = size(a);
t = input("Enter the threshold parameter: ");
b = zeros(m, n);  // Initialize b with zeros
for i = 1:m
   for j = 1:n
      if a(i,j) < t
         b(i,j) = 0;
      else
         b(i,j) = 255;
      end
   end
end
figure(1);
imshow(uint8(a));
title('Original Image');
figure(2);
imshow(b);
title('Thresholded Image');
imwrite(b, "C:\Users\admin\Desktop\output_threshold.jpg" );
```

Output-

Original Image-

Modified Image-



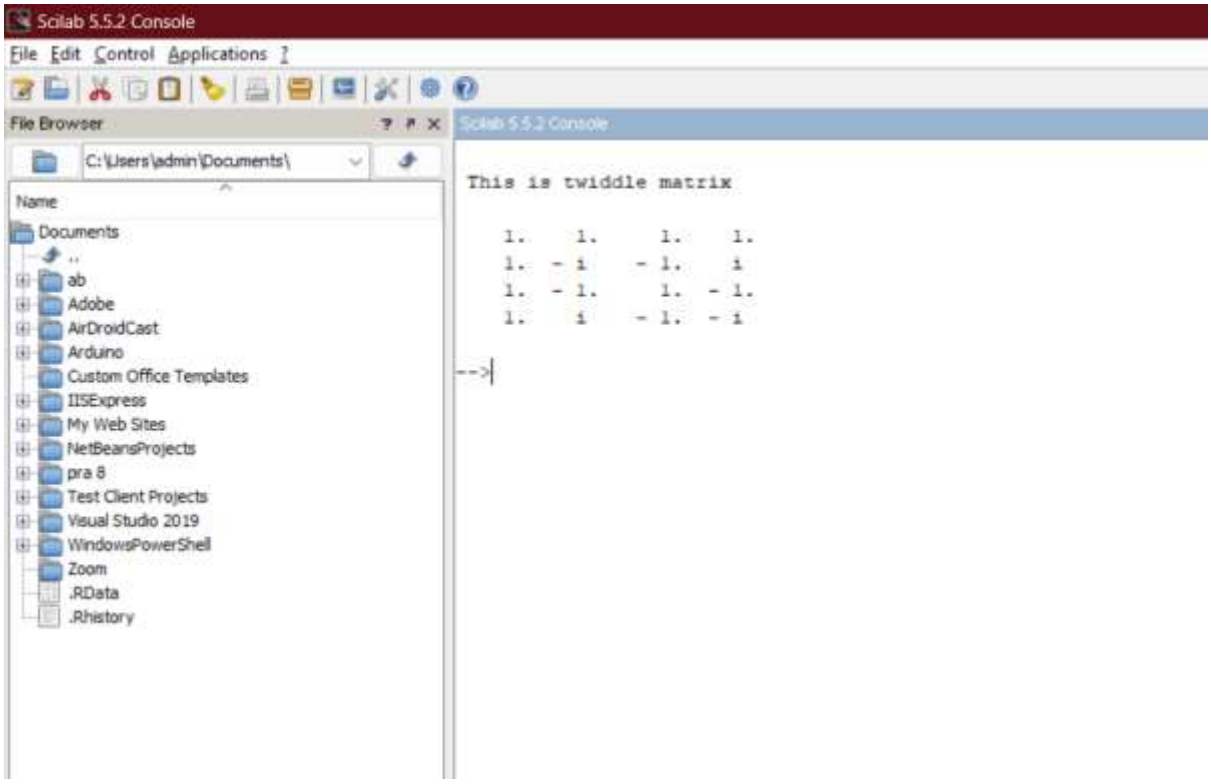
Practical 8

Aim- create a twiddle matrix of size N x N, where N=4. The twiddle matrix is a matrix of complex exponential values used in the computation of the discrete Fourier transform (DFT).

Code-

```
clc;
N=4
for i=0:N-1
   for j=0:N-1
      W(i+1,j+1)=int(cos(2*%pi*i*j/N)-%i*sin(2*%pi*i*j/N));
   end
end
disp('This is twiddle matrix')
disp(W)
```

Output-



Practical 9

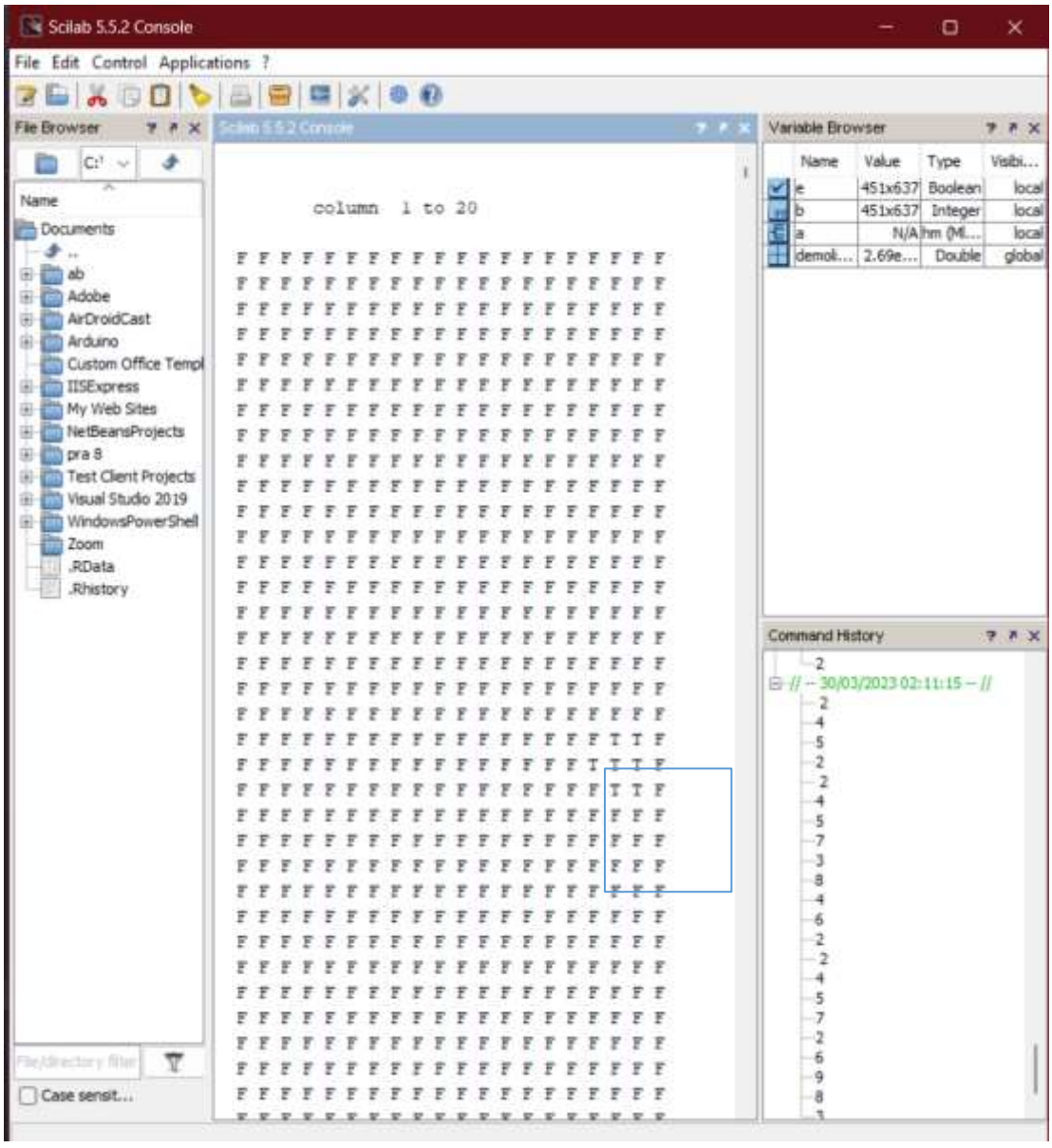
Aim- To apply the Sobel operator to detect edges, and displays the resulting edge map.

Code-

```
clc;
clear;
close();
a=imread("C:\Users\admin\Desktop\dip\22f5b15c-396b-4200-aba9-6946cc097d33.jfif")
b=rgb2gray(a);
e=edge(b,"sobel");
disp(e)
```

Ouput-

The box shows the edge detected.



Practical 10

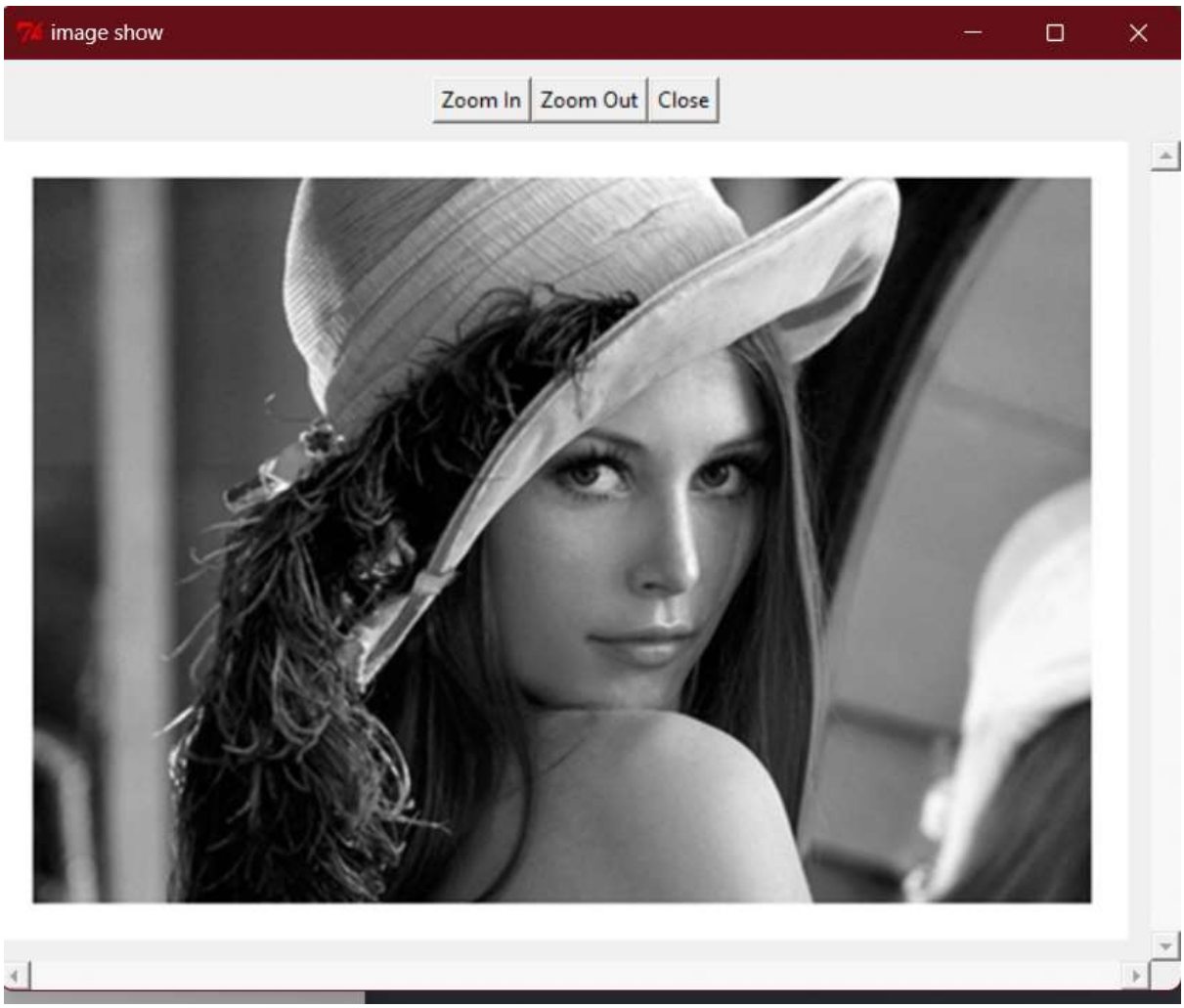
Aim- perform Canny edge detection on a grayscale image and display the resulting edge map using SCILAB.

Code-

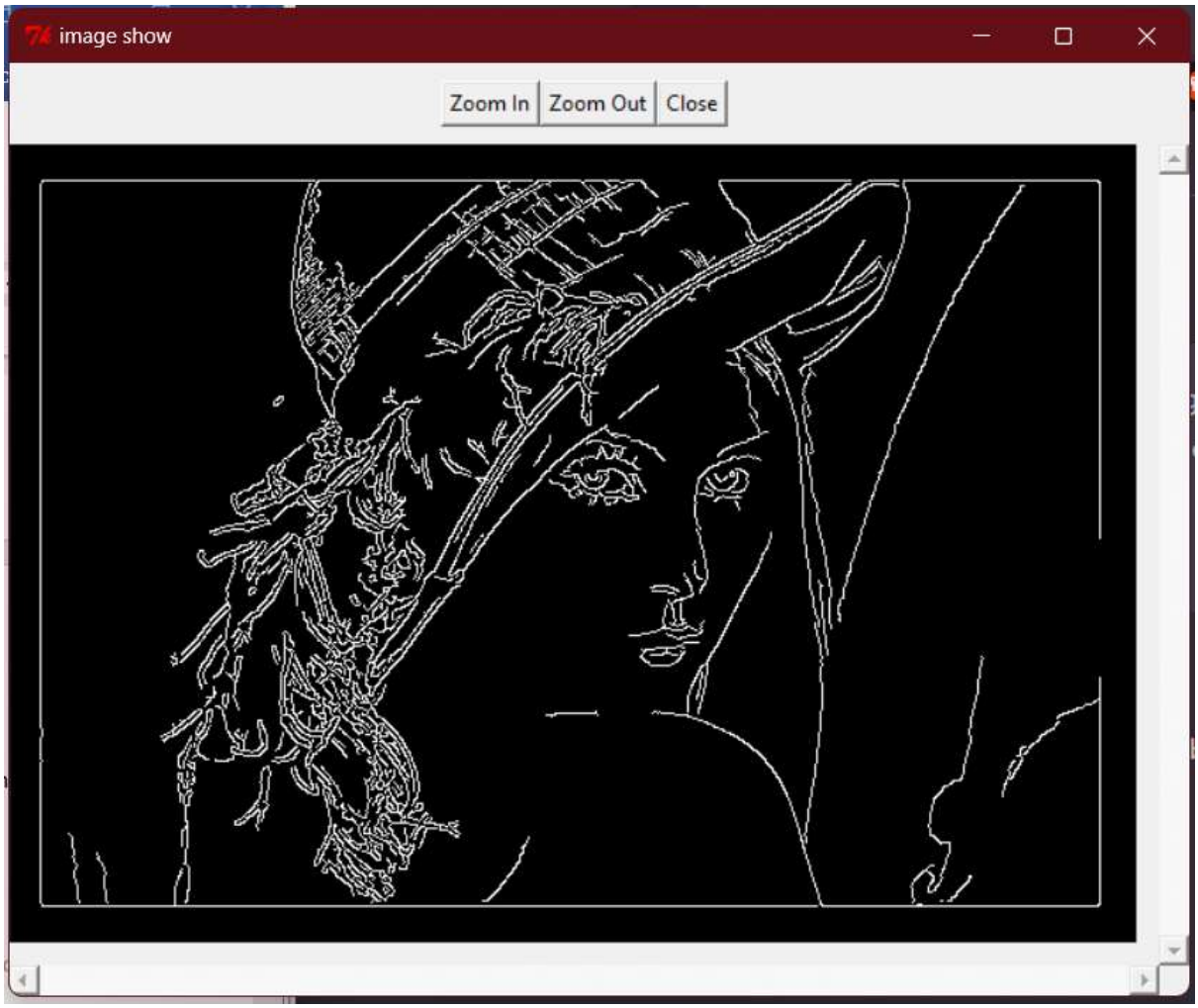
```
clc;
xdel(winsid())
a=imread("C:\Users\admin\Desktop\dip\22f5b15c-396b-4200-aba9-6946cc097d33.jfif")
a=rgb2gray(a)
e=edge(a,'canny')
figure(1)
imshow(e)
```

Output-

Original Image-



Modified Image-



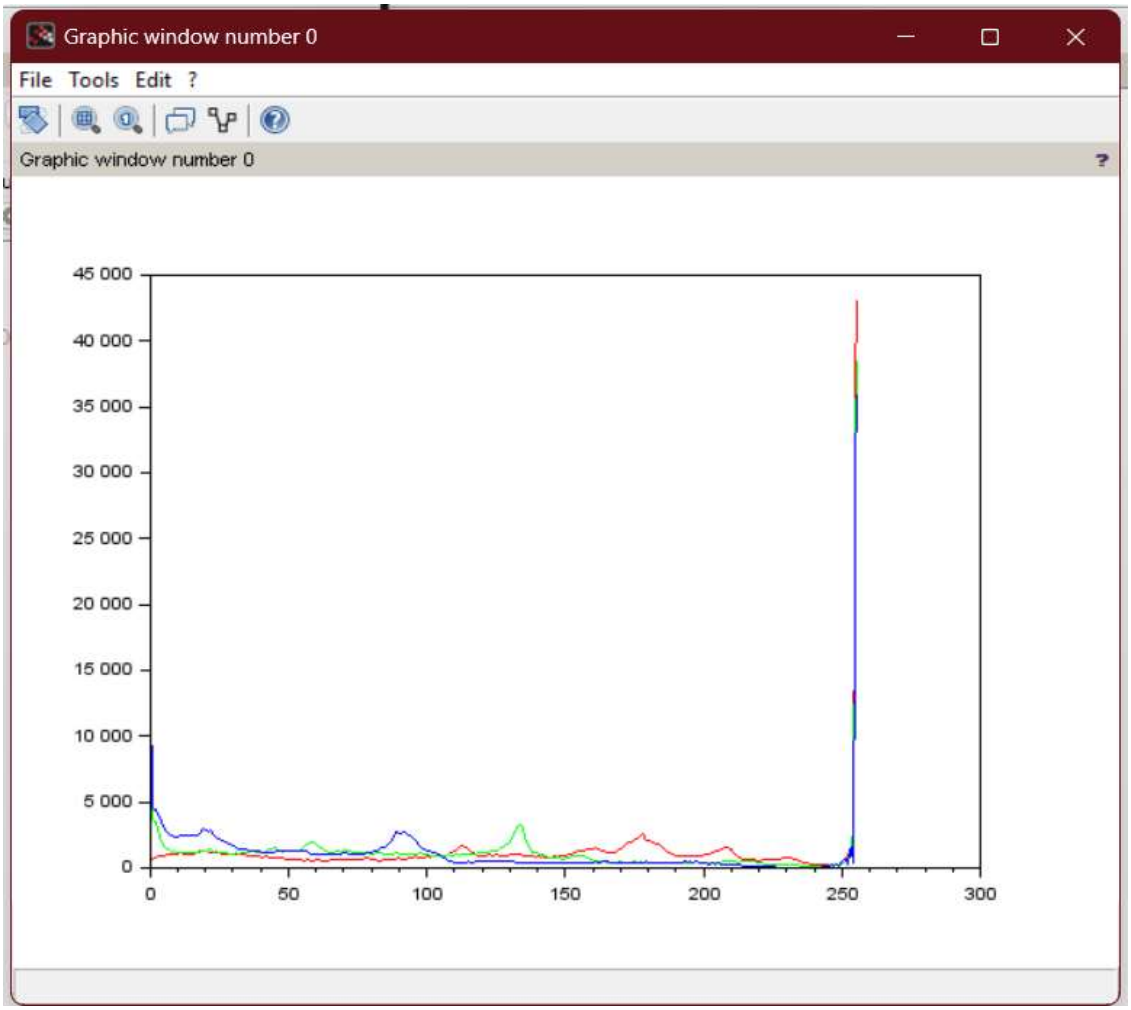
Practical 11

Aim- perform edge detection on a grayscale image using the Canny edge detection algorithm and display the resulting image using a histogram.

Code-

```
clc;
clear;
image=imread("C:\Users\admin\Desktop\dip\22f5b15c-396b-4200-aba9-6946cc097d33.jfif")
red=image(:,:,1)
green=image(:,:,2)
blue=image(:,:,3)
[yRed,x]=imhist(red)
[yGreen,x]=imhist(green)
[yBlue,x]=imhist(blue)
plot(x,yRed,'Red',x,yGreen,'Green',x,yBlue,'Blue')
```

Output-



Practical 12

Aim- perform gray-level slicing on an input image, which involves creating a binary image based on a specified range of gray levels in the original image.

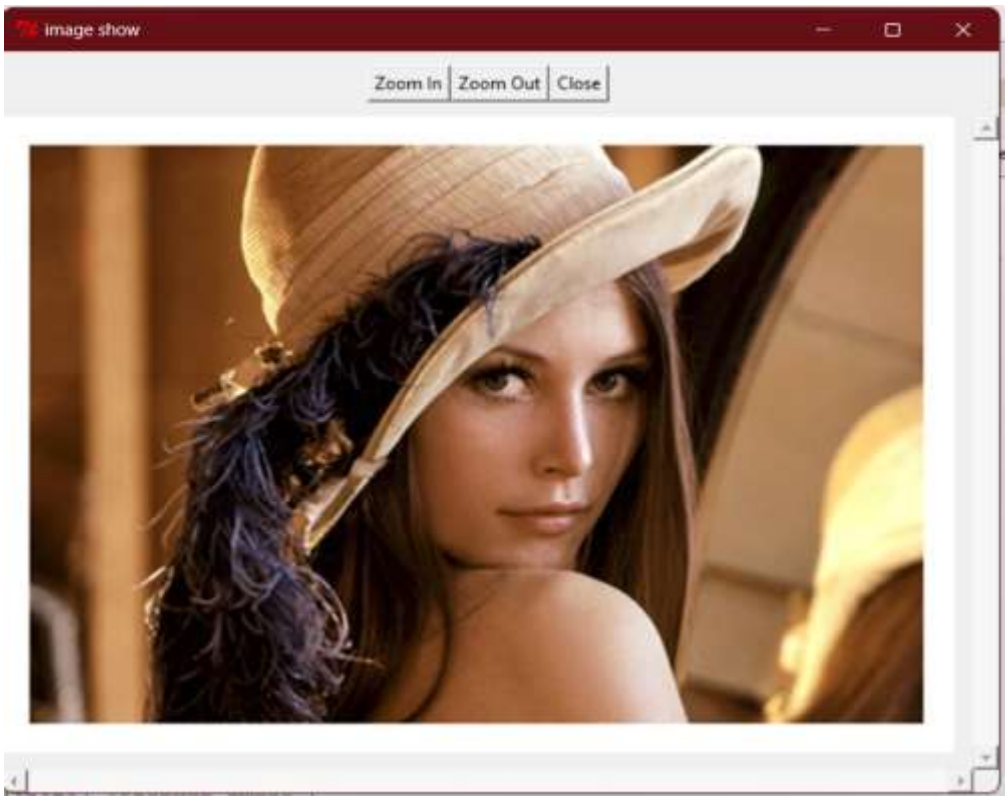
Code-

```
clc;
//disp('for first matrix')
x=imread("E:\\Cameraman.jpg")
y=double(x);

[m,n]=size(y)
L=max(x);
a=round(L/2);
b=L;
for i=1:m
   for j=1:n
      if(y(i,j)>=a & y(i,j)<=b)
         z(i,j)=255;
      else
         z(i,j)=0;
      end
   end
end
figure(1);
imshow(uint8(x));
title('original image')
figure(2);
imshow(uint8(z));
title('threshold image')
```

Output-

Original Image-



Modified Image-