Assessment No. 01

AIM: 2D Linear Convolution, Circular Convolution between two 2D Matrices.

CODE:

(1) Linear Convolution

CODE:

```
x=[1 2;3 4]
h=[5 6;7 8]
y=conv2(x,h);
disp("Linear convolution is y(m,n)",y);
```

OUTPUT:

```
--> exec('C:\Users\admin\Desktop\DIP\s5.sce', -1)

  5.    16.   12.
  22.   60.   40.
  21.   52.   32.

 Linear convolution is y(m,n)
```

(2) Circular Convolution

CODE:

```
x=[1 2; 3 4]
h=[5 6; 7 8]
X=fft2(x)
H=fft2(h)
y=X.*H
Y=ifft(y)
disp(Y)
```

OUTPUT:

```
--> exec('C:\Users\admin\Desktop\DIP\s4.sce', -1)

  70.   68.
  62.   60.
```

CONCLUSION: We have successfully performed 2D Linear Convolution, Circular Convolution using Scilab.

Assessment No. 02

AIM: Circular Convolution expressed as linear convolution plus alias.

CODE:

```
x=[1 2;3 4]
h=[5 6;7 8]
y=conv2(x,h);
disp("Linear convolution is y(m,n)",y);
y1=[y(:,1)+y(:,$),y(:,2)];
y2=[y1(1,:)+y1($,:);y1(2,:)];
disp("Circular convolution is y(m,n)",y2);
```

OUTPUT:

```
--> exec('C:\Users\admin\1.sce', -1)

   5.    16.   12.
   22.   60.   40.
   21.   52.   32.

 Linear convolution is y(m,n)

   70.   68.
   62.   60.

 Circular convolution is y(m,n)
```

CONCLUSION: We have successfully performed Circular Convolution using Scilab.

Assessment No. 03

AIM: Linear correlation of a 2D matrix, Circular correlation between two signals and Linear auto correlation of a 2D matrix.

1. Linear correlation of a 2D matrix

Code:

```
clc;
x1=[31;24]
x2=[15;23]
n=x2(:,$:-1:1)
m=n($:-1:1,:)
disp(n);
disp(m);
y=conv2(x1,m);
disp("Linear correlation is y(m,n)",y);
```

OUTPUT:

```
--> n=x2(:,$:-1:1)
 n  =

   5.   1.
   3.   2.
--> m=n($:-1:1,:)
 m  =

   3.   2.
   5.   1.
--> disp(n);

   5.   1.
   3.   2.
--> disp(m);

   3.   2.
   5.   1.
--> y=conv2(x1,m);
--> disp("Linear correlation is y(m,n)",y);

   9.    9.    2.
   21.   24.   9.
   10.   22.   4.

 Linear correlation is y(m,n)
```

CONCLUSION: We have successfully performed Linear correlation using Scilab.

2. Circular correlation between two signals.

CODE:

```
clc;
x1=[15;24];
x2=[32;41];
h=x2(:,$:-1:1);//left to right direction
h1=h($:-1:1,:);//up to down direction
x=fft(x1);
h2=fft(h1);
y=ifft(x.*h2);
disp(y,circularcorrelation);//circular correlation
```

OUTPUT:

```
 circular correlation

   37.   23.
   35.   25.
-->
```

CONCLUSION: We have successfully performed Circular correlation using Scilab.

3. Linear Auto correlation of a 2D matrix.

CODE:

```
x1=[32;15];
x2=x1(:,$:-1:1);//left to right direction
h=x2($:-1:1,:);//up to down direction
y=conv2(x1,h);
disp(y,"linear auto correlation");//linear auto correlation
```

OUTPUT:

```
linear auto correlation


  15.   13.   2.
  11.   39.   11.
  2.    13.   15.
```

CONCLUSION: We have successfully performed linear auto correlation using Scilab.

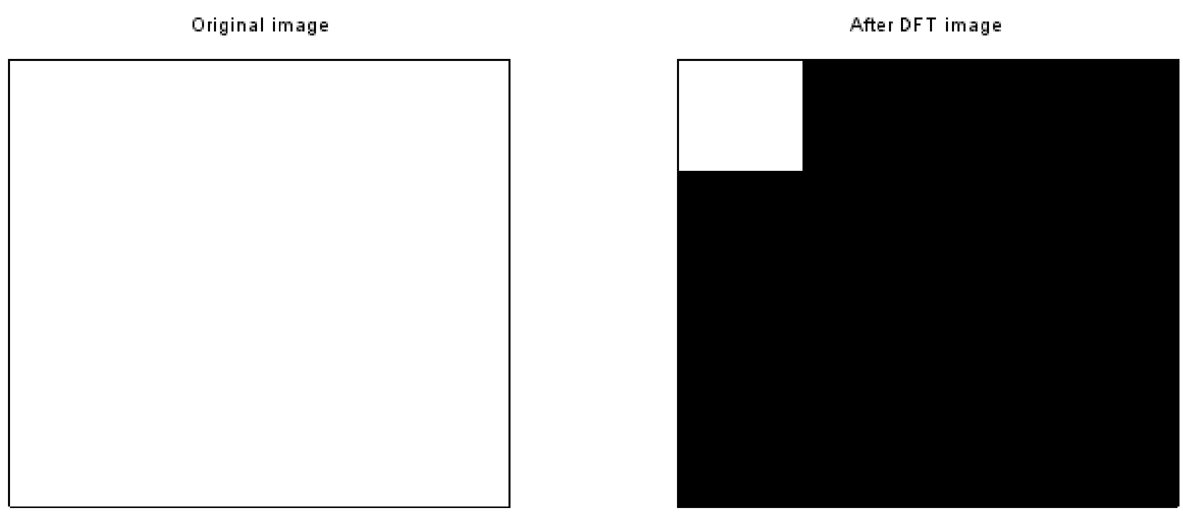
Assessment No. 04

AIM: Compute 2D DFT of 4*4 Grayscale image.

CODE:

```
f=[1 1 1 1;1 1 1 1;1 1 1 1;1 1 1 1];
N=4;
kernel=fft(N);
F=kernel*(f*kernel ');
F1=ifft(F);
disp(F1);
subplot(221);
imshow(f);
title('Original image');
subplot(222);
imshow(F1);
title('After DFT image');
```

OUTPUT:




```
  16.   0.   0.   0.
  0.    0.   0.   0.
  0.    0.   0.   0.
  0.    0.   0.   0.
-->
```

2) Compute the inverse of 2D DFT of the transform coefficient given by

CODE:

```
f=[16 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
N=4;
kernel=fft(N);
f=kernel*(f*kernel')/N^2
F1=ifft(f);
disp(F1);
```

OUTPUT:

```
--> disp(F1);

   1.   1.   1.   1.
   1.   1.   1.   1.
   1.   1.   1.   1.
   1.   1.   1.   1.
-->
```

CONCLUSION: We have successfully implemented DFT on 4*4 grayscale matrix using Scilab

Assessment No. 05

AIM: Compute discrete cosine transform, Program to perform KL Transform for the given 2D matrix.

1.Compute discrete cosine transform

(i) Compute the discrete cosine transform (DCT) matrix for N=4

CODE:

```
clc;
N=input('Enter the length of DCT matrix');
const=sqrt(2/N);
for k=0:1:N-1
   for l=0:1:N-1
   if k==0 then
      c(k+1,l+1)=sqrt(1/N);
   else
      c(k+1,l+1)=const*cos(((2*l+1)*%pi*k)/(2*N));
end
   end
end
disp(c);
```

OUTPUT: DCT matrix of order four

```
0.5         0.5         0.5         0.5
0.6532815   0.2705981  -0.2705981  -0.6532815
0.5        -0.5        -0.5         0.5
0.2705981  -0.6532815   0.6532815  -0.2705981
```

(ii) Computation of the DCT of a given matrix (sequence).

CODE:

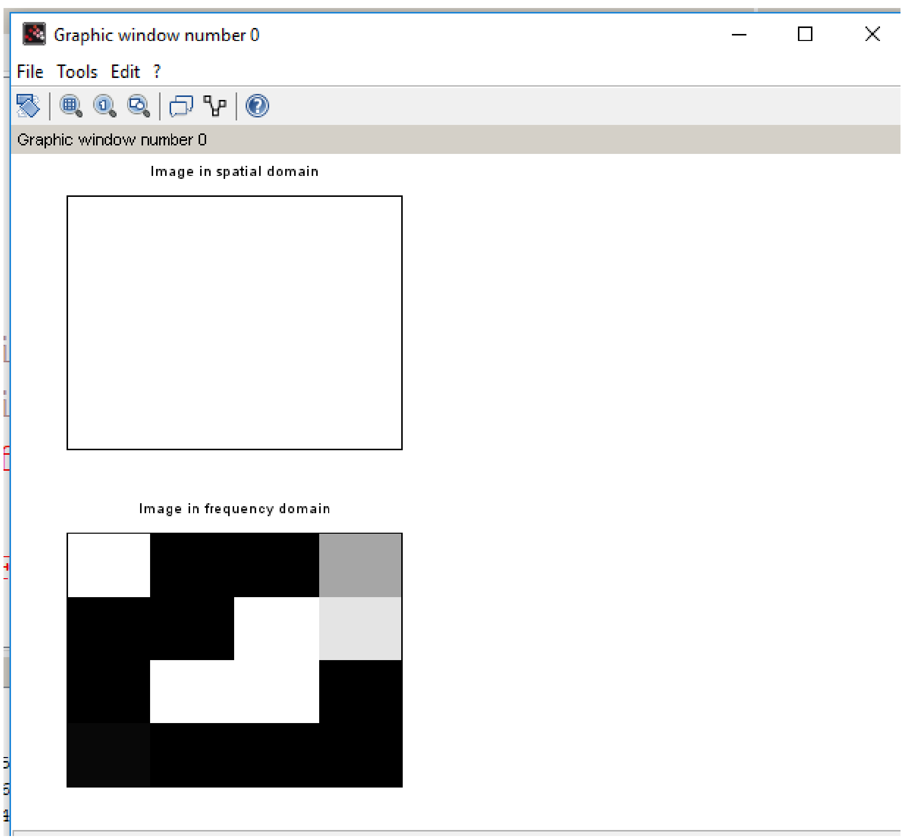
```
clc;
f=[2 4 4 2;4 6 8 3;2 8 10 4;3 8 6 2];
[M N]=size(f);
const=sqrt(2/N);
for k=0:1:N-1
   for l=0:1:N-1
       if k==0
          c(k+1,l+1)=1/sqrt(N);
       else
          a=2*l;
       c(k+1,l+1)=const*cos((%pi*k*(a+1))/(2*N));         end
  end
end
f1=c*f;
disp(f1);
F=c*f*c ';
disp(c,'Discrete Cosine Transform');
disp(F,'Discrete Cosine Transform of f(m,n) is');
subplot(221);imshow(f);title('Image in spatial domain')
subplot(223);imshow(F);title('Image in frequency domain')
```

<u>Another code for same program using DCT() function</u>

```
clc;
f=[2 4 4 2;4 6 8 3;2 8 10 4;3 8 6 2];
F=dct(f);
disp(F);
subplot(221);imshow(f);title('Image in spatial domain')
subplot(223);imshow(F);title('Image in frequency domain')
```

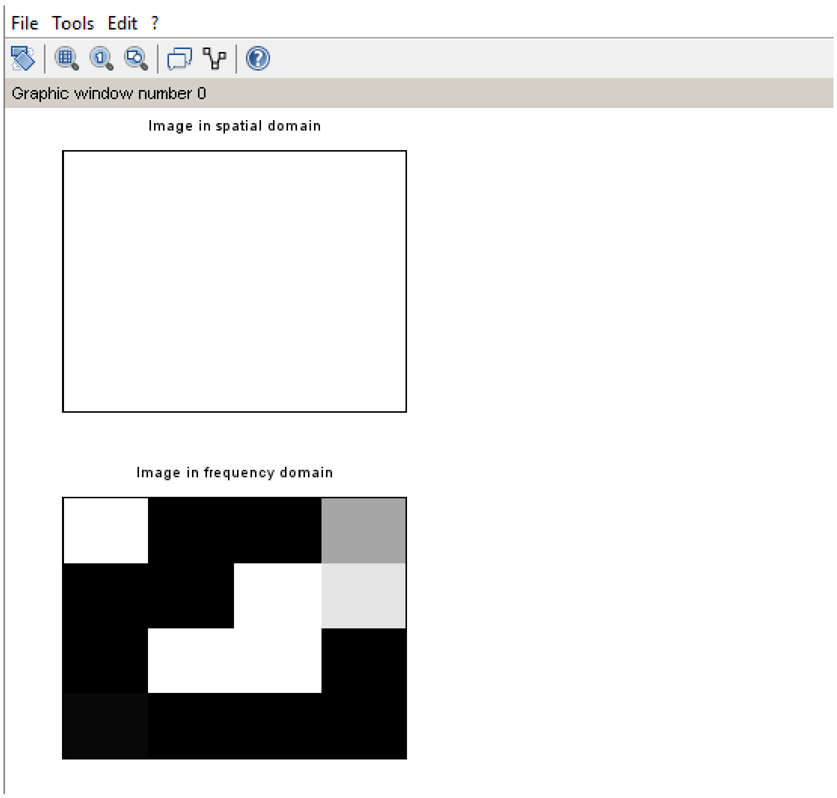
OUTPUT:

```
   5.5          13.          14.          5.5
  -0.1120854   -3.154322    -1.8477591   -0.2705981
  -0.5         -1.          -4.          -1.5
  -1.577161     0.2241708    0.7653669    0.6532815
--> F=c*f*c ';
--> disp(c,'Discrete Cosine Transform');

 Discrete Cosine Transform

   0.5          0.5          0.5          0.5
   0.6532815    0.2705981   -0.2705981   -0.6532815
   0.5         -0.5         -0.5          0.5
   0.2705981   -0.6532815    0.6532815   -0.2705981
--> disp(F,'Discrete Cosine Transform of f(m,n) is');

 Discrete Cosine Transform of f(m,n) is

   19.         -0.2705981   -8.           0.6532815
  -2.6923823   -0.25         2.3096988    0.8964466
  -3.5          1.4650756    1.5         -1.6892464
   0.032829    -1.6035534   -0.9567086   -0.25
```



Using DCT function

```
--> F=dct(f);
--> disp(F);

   19.         -0.2705981  -8.          0.6532815
  -2.6923823   -0.25        2.3096988   0.8964466
  -3.5          1.4650756   1.5        -1.6892464
   0.032829    -1.6035534  -0.9567086  -0.25
--> subplot(221);imshow(f);title('Image in spatial domain')
--> subplot(223);imshow(F);title('Image in frequency domain')
-->
```



Program to perform KL transform for the given 2D matrix.

Code:

```
clc;
//X=[4 3 5 6;4 2 7 7;5 5 6 7];
X=input('Enter the matrix');
[m,n]=size(X);
A=0;
E=0;
for i=1:n
   A=A+X(:,i);
   E=E+X(:,i)*X(:,i)';
end
mx=A/n;
E=E/n;
C=E-mx*mx';
[V,D]=spec(C);
d=diag(D);
[d,i]=gsort(d);
for j=1:length(d)
   T(:,j)=V(:,i(j));
end
T=T'
disp(d,'Eigen values are U=')
disp(T,'The Eigen vector matrix T=')
disp(T,'The KL transform basis is =')
for i=1:n
   Y(:,i)=T*X(:,i);
end
disp(Y,'KL transformation of the input matrix Y=');
for i=1:n
   x(:,i)=T' *Y(:,i);
end
disp(x,'Reconstruct matrix of the given sample matrix X= ')
```

Output:

```
Scilab 6.0.1 Console
Enter the matrix[4 3 5 6;4 2 7 7;5 5 6 7]


 Eigen values are U=

   6.1963372
   0.2147417
   0.0264211

 The Eigen vector matrix T=

   0.4384533   0.8471005   0.3002988
   0.4460381  -0.4951684   0.7455591
  -0.780262    0.1929481   0.5949473

 The KL transform basis is =

   0.4384533   0.8471005   0.3002988
   0.4460381  -0.4951684   0.7455591
  -0.780262    0.1929481   0.5949473

 KL transformation of the input matrix Y=

   6.6437095   4.5110551   9.9237632   10.662515
   3.5312743   4.0755729   3.2373664   4.4289635
   0.6254808   1.0198466   1.0190104   0.8336957

 Reconstruct matrix of the given sample matrix X=

   4.   3.   5.   6.
   4.   2.   7.   7.
   5.   5.   6.   7.
```

CONCLUSION: We have successfully implemented Compute discrete cosine transform & KL Transform for given 2D transform using Scilab.

Assessment No. 6

AIM: i) Brightness Enhancement of image

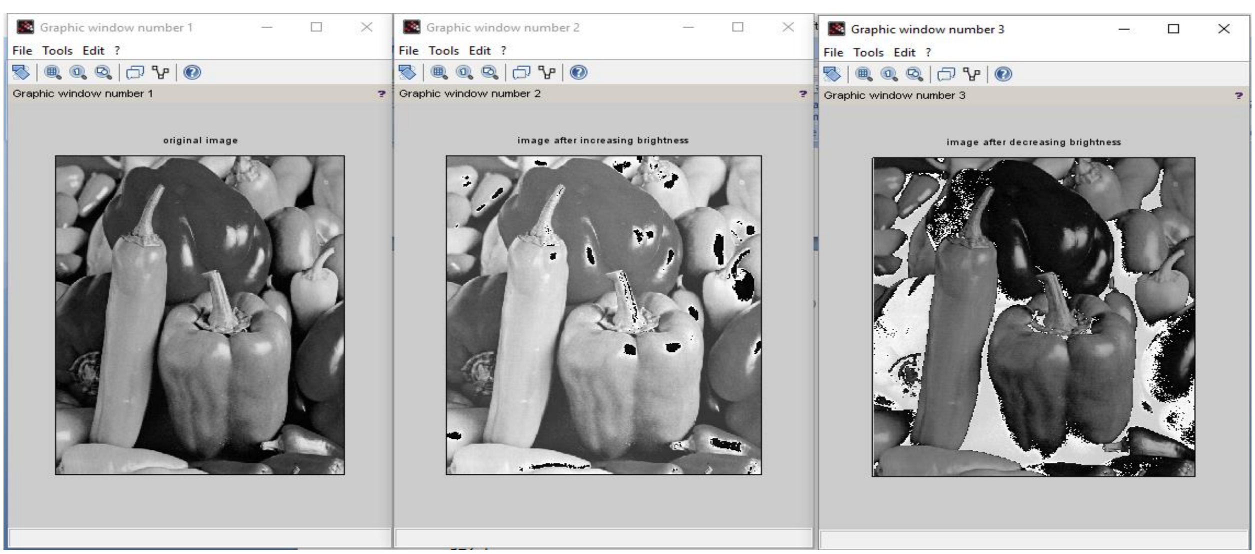
ii) Contrast Manipulation

iii) Image Negative

CODE:

```
clc;
k=input('Enter the constant value')
x=imread(fullpath(getIPCVpath()+"images/peppers.png"));
a=rgb2gray(x);
g=double(a)+k;//Increasing the brightness of the image by constant k
g1=uint8(g);
d=double(a)-k;
d1=uint8(d);
figure(1);
imshow(a);
title('original image')
figure(2);
imshow(g1);
title('image after increasing brightness')
figure(3);
imshow(d1);
title('image after decreasing brightness')
```

Output:



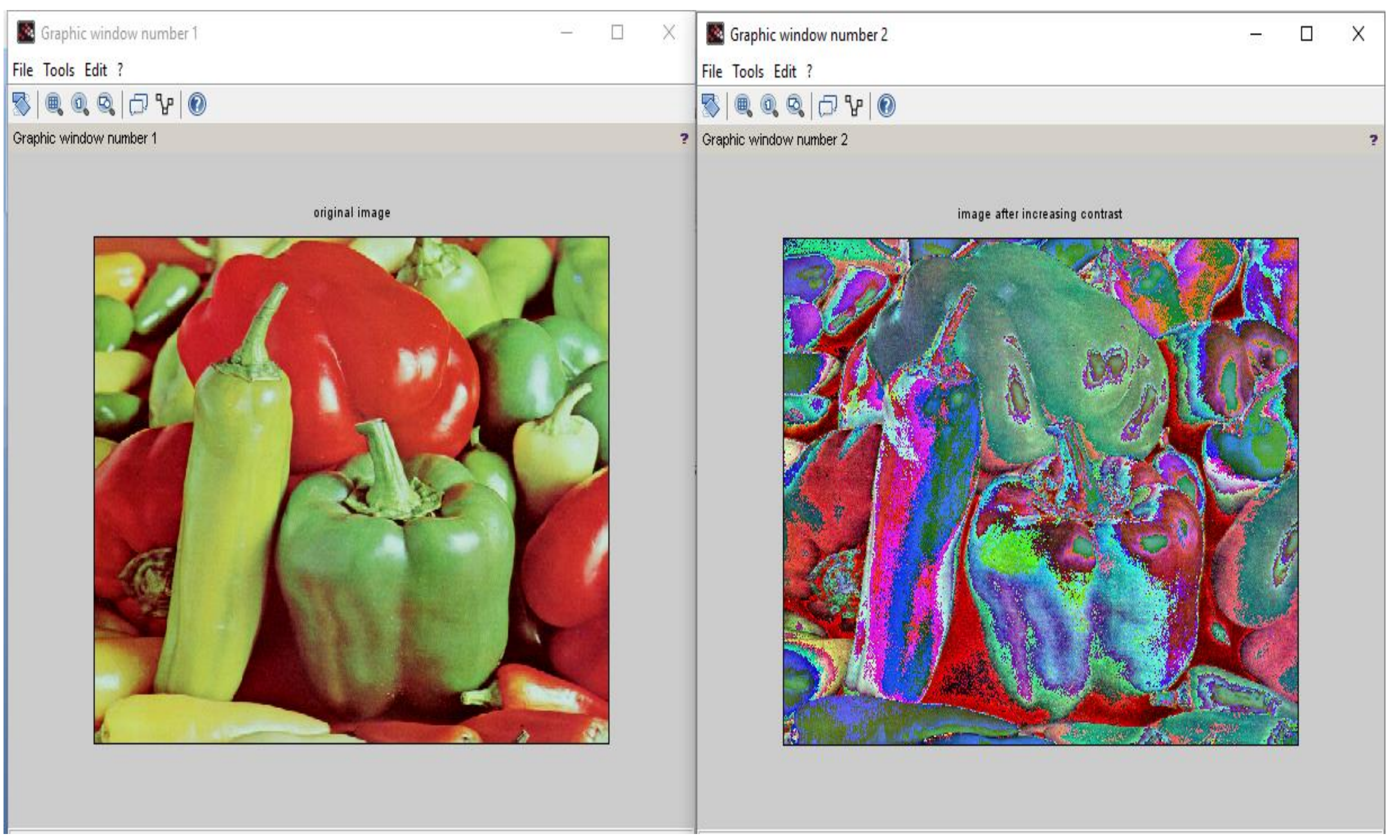
Code: Contrast Manipulation

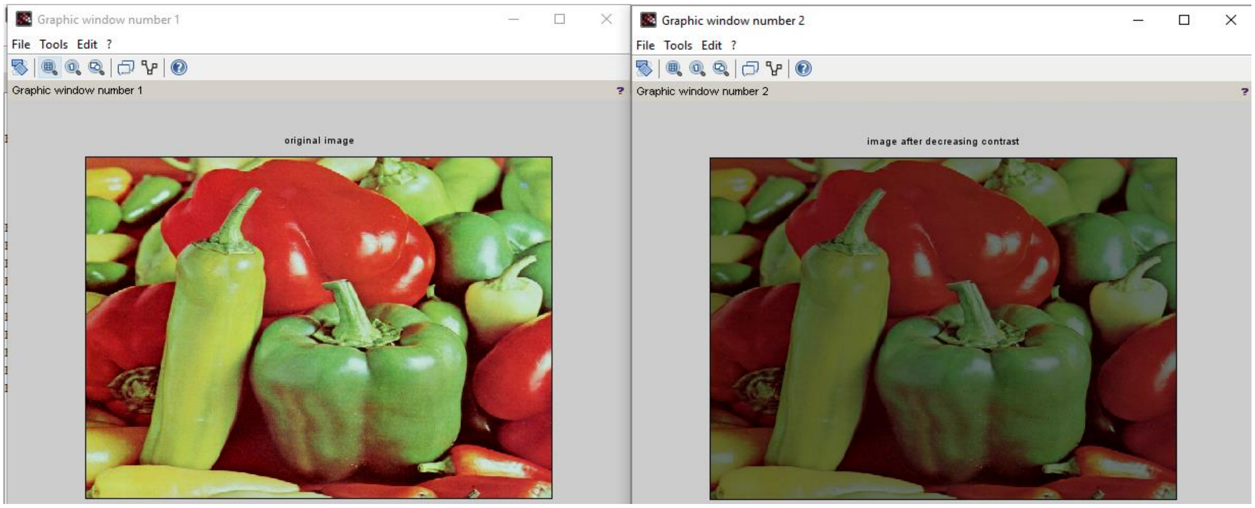
Take k<1 for increasing contrast:

Take k>1 for decreasing contrast:

```
clc;
k=input('Enter the constant value k')
x=imread(fullpath(getIPCVpath()+"images/peppers.png"));
a=(x);
g=double(a)*k;//Increasing the contrast of the image by constant k
g1=uint8(g);
figure(1);
imshow(a);
title('original image')
figure(2);
imshow(g1);
title('image after increasing contrast')
```

Output:

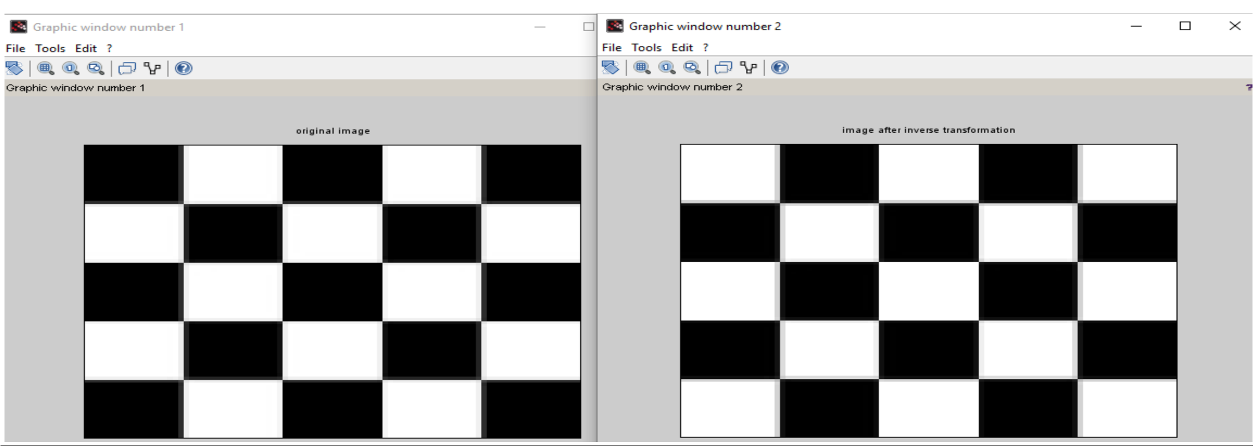




Code: Image Negative:

```
clc;
x=imread(fullpath(getIPCVpath()+"images/checkerbox.png"));
a=(x);
g=255-double(a);//Increasing the contrast of the image by constant k
g1=uint8(g);
figure(1);
imshow(a);
title('original image')
figure(2);
imshow(g1);
title('image after inverse transformation')
```

Output:



Conclusion: We have successfully studied brightness enhancement, contrast manipulation and reverse transformation of an image.

Assessment No. 07

AIM: i) Perform threshold operation

ii)Perform Graylevel slicing without Background

Code: Without Background & with background

```
clc;
clear all;
y=imread(fullpath(getIPCVpath()+"/images/peppers.png"));
//x1=rgb2gray(x);
x2=double(y);
[m n]=size(x2);
L=double(255);
c=double(round(L/1.25));
b=double(round(2*L/2));
for i=1:m
   for j=1:n
      if(x2(i,j)>=c & x2(i,j)<=b)
         z(i,j)=L;
      else
         z(i,j)=0; for with background change z(i,j)=0 to z(i,j)=x2
      end
   end
end
z=uint8(z);
subplot(221);
imshow(y);
title('original image')
subplot(222);
imshow(z);
title('Gray level slicing without background image')
```

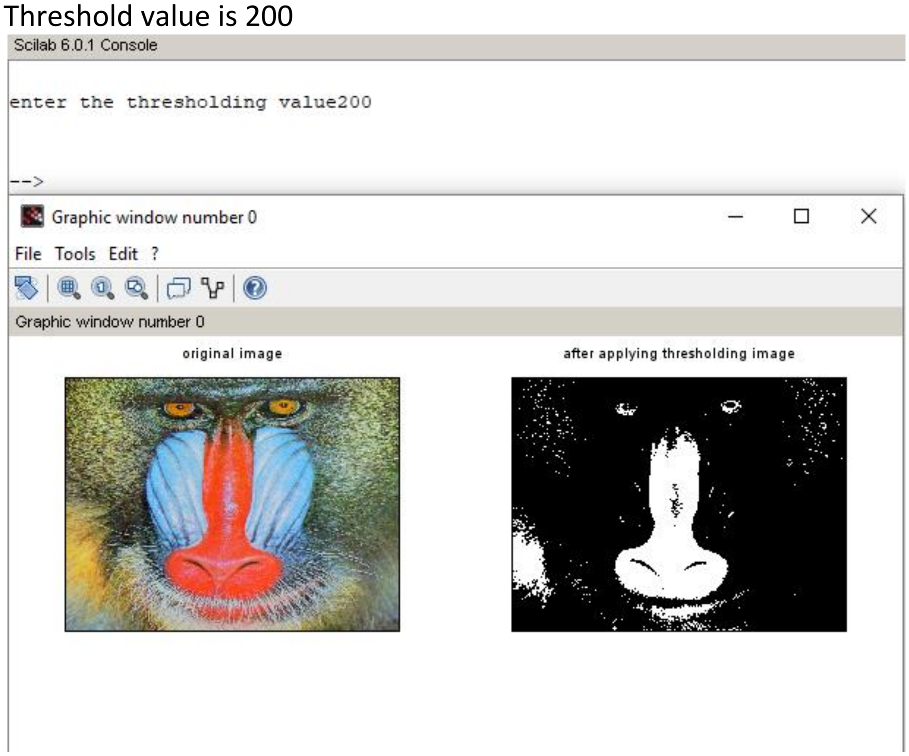
Output:

Code: Perform threshold operation

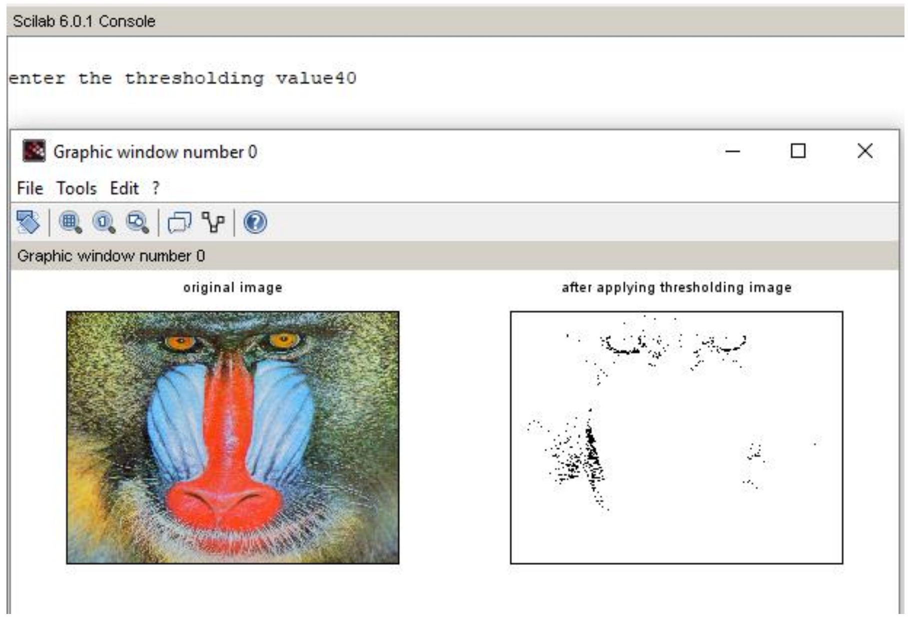
```
//program to perform thresholding
clc;
y=imread(fullpath(getIPCVpath()+"/images/peppers.png"));
[m n]=size(y);
t=input('enter the thresholding value');
for i=1:m
   for j=1:n
      if(y(i,j)<t)
         z(i,j)=0
      else
         z(i,j)=255
      end
   end
end
subplot(221);
imshow(y);
title('original image')
subplot(222);
imshow(z);
title('after applying thresholding image')
```

Output:

Threshold value is 200



Threshold value is 40



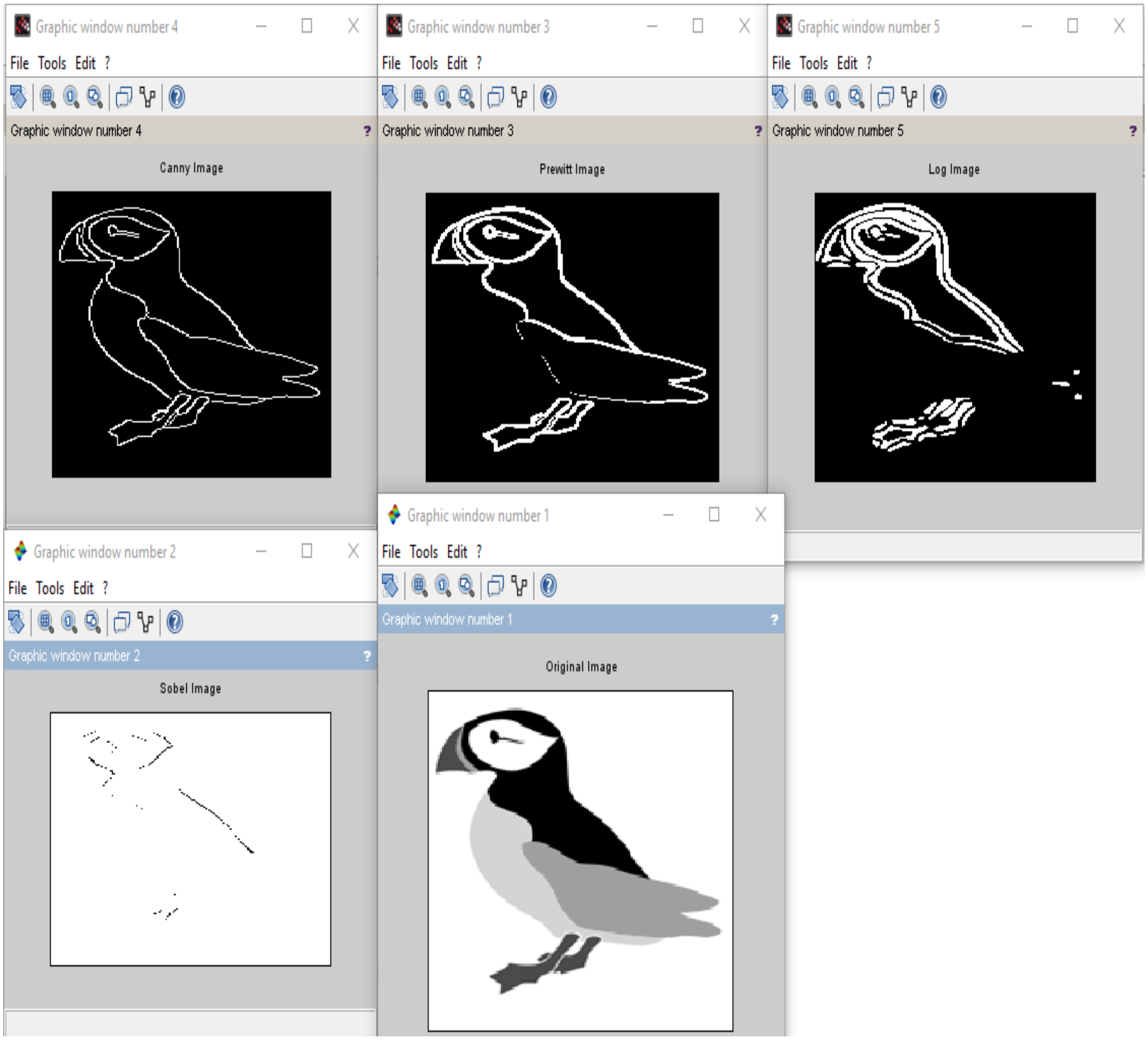
<u>Conclusion:</u> We have successfully studied Threshold operation.

Assessment No. 08

AIM: Image Segmentation

Code:

```
clc;
im=imread(fullpath(getIPCVpath()+"images/puffin.png"));
im=rgb2gray(im);
s=edge(im,'sobel');
p=edge(im,'prewitt');
l=edge(im,'log');
c=edge(im,'canny');
figure(1);
imshow(im);
title('Original Image');
figure(2);
imshow(s);
title('Sobel Image');
figure(3);
imshow(p);
title('Prewitt Image');
figure(4);
imshow(c);
title('Canny Image');
figure(5);
imshow(l);
title('Log Image');
```



<u>Conclusion:</u> We have successfully studied Image Segmentation.

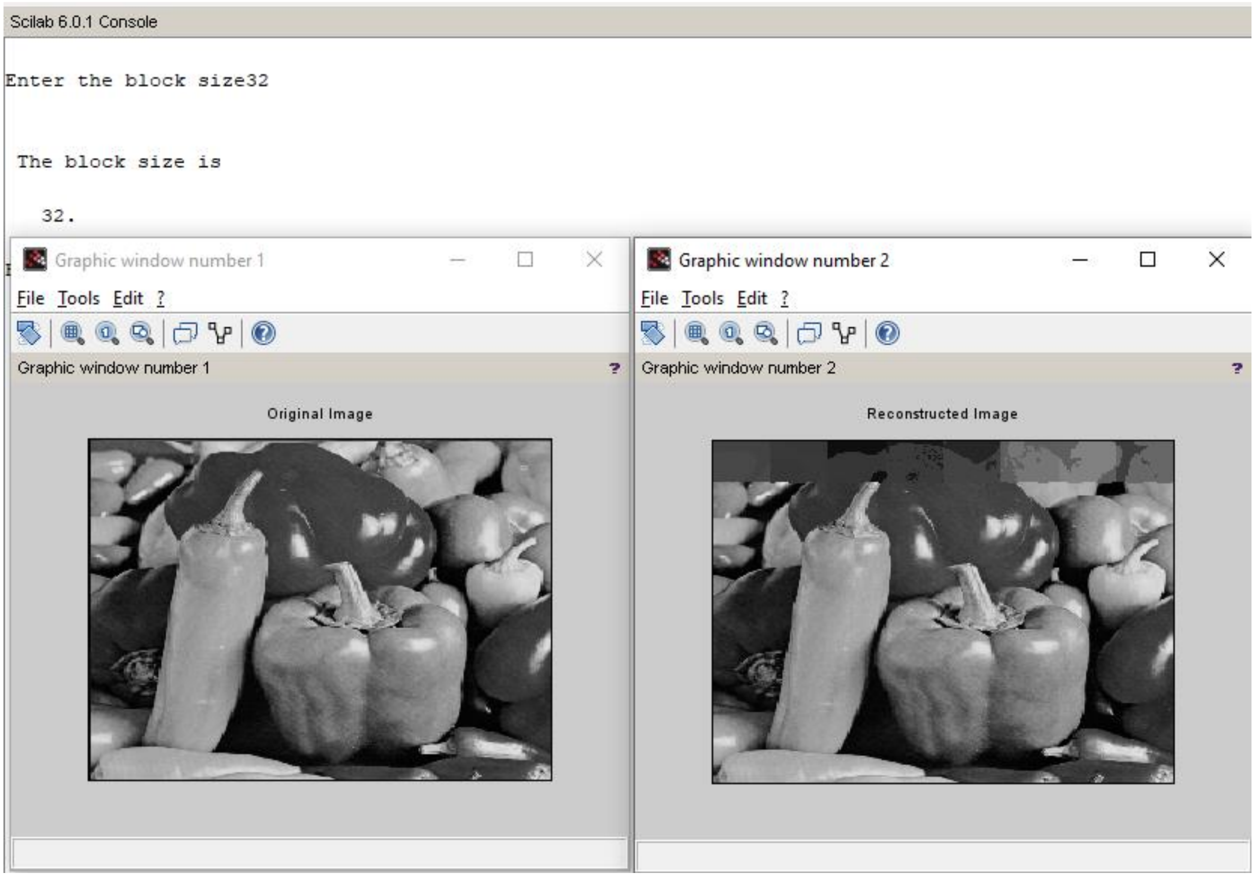
Assessment No. 09

AIM: Image Compression

Code: Perform Compression

```
clc;
x=imread(fullpath(getIPCVpath()+"images/peppers.png"));
x=rgb2gray(x);
x=imresize(x,[256 256]);
x=double(x);
x1=x;
[m n]=size(x);
blk=input('Enter the block size');
for i=1:blk:m
   for j=1:blk:n
       y=x(i:1+(blk-1),j:j+(blk-1));
       m1=mean(mean(y));
       sig=std2(y)
       b=y>m1;
       K=sum(sum(b));
          if(K~=blk^2)&(K~=0)
             m=m1-sig*sqrt(K/((blk^2)-K))
             mu=m1-sig*sqrt(((blk^2)-K)/K)
             x(i:1+(blk-1),j:j+(blk-1))=b*mu+(1-b)*m
          end
    end
end
figure(1)
imshow(uint8(x1))
title('Original Image')
figure(2)
imshow(uint8(x))
title('Reconstructed Image')
disp(blk,'The block size is')
```

Output:


Scilab 6.0.1 Console
Enter the block size32
The block size is
32.
Graphic window number 1
File Tools Edit ?
Graphic window number 1
Original Image
Graphic window number 2
File Tools Edit ?
Graphic window number 2
Reconstructed Image

Assessment No. 10

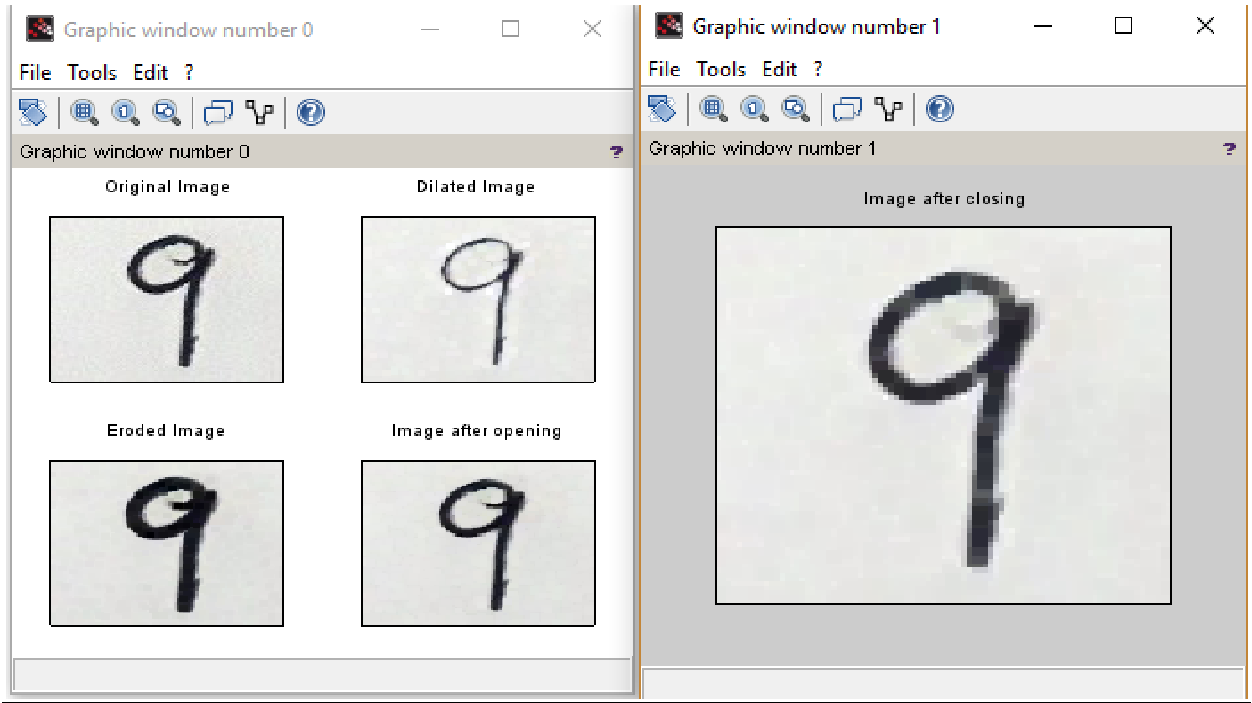
AIM: i) Binary Image Processing

ii) Color Image Processing

Code (Binary Image Processing):

```
clc;
X=imread(fullpath(getIPCVpath()+"images/dnn/5.jpg"));
B=[1 1 1;1 1 1;1 1 1];
im1=imdilate(X,B);
im2=imerode(X,B);
im3=imopen(X,B);
im4=imclose(X,B);
subplot(221);
imshow(X);
title('Original Image');
subplot(222);
imshow(im1);
title('Dilated Image');
subplot(223);
imshow(im2);
title('Eroded Image');
subplot(224);
imshow(im3);
title('Image after opening');
figure(1);
imshow(im4);
title('Image after closing');
```

Output:



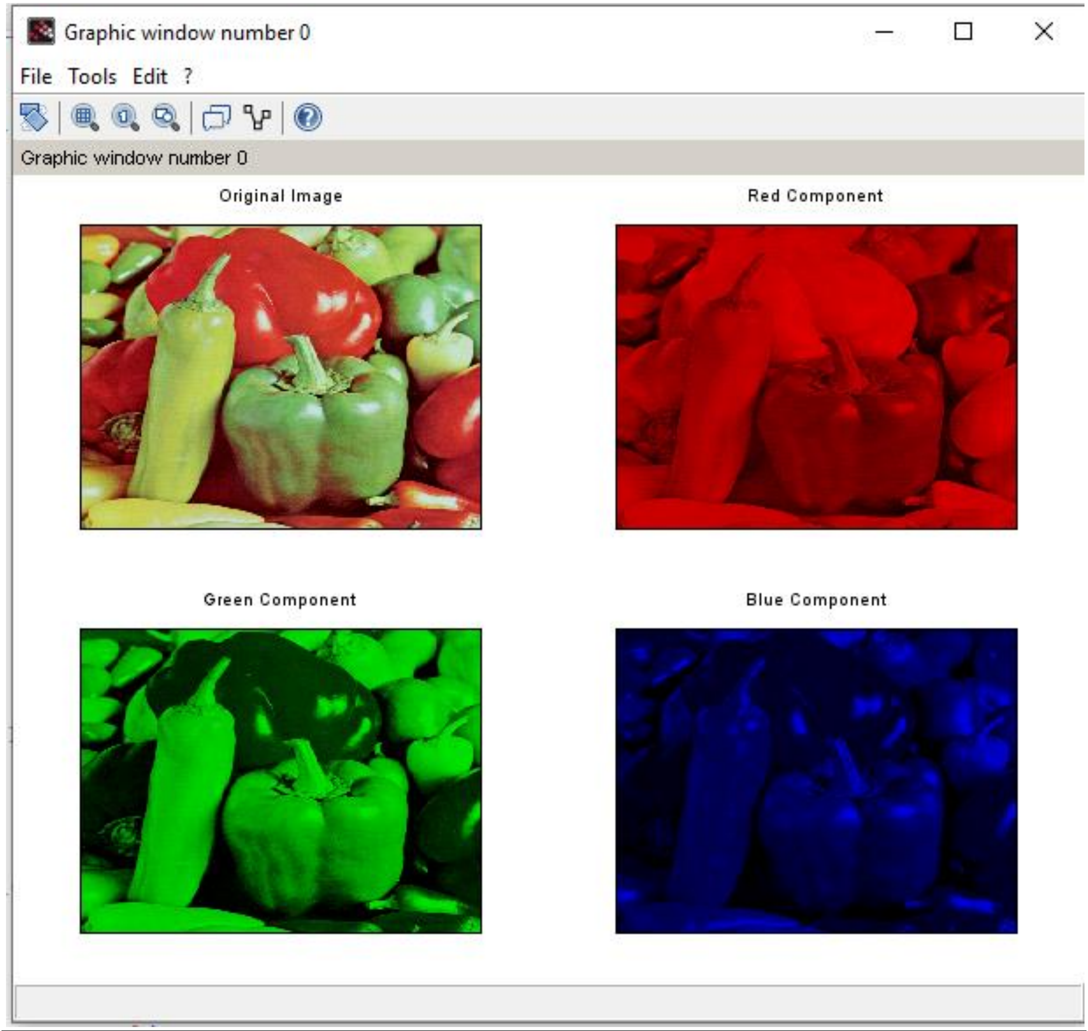
Code (Color Image Processing):

1) Program to read RGB image and extract b3 color component.

```
clc;
RGB=imread(fullpath(getIPCVpath()+"images/peppers.png"));
R=RGB;
G=RGB;
B=RGB;
R(:,:,2)=0;
R(:,:,3)=0;
G(:,:,1)=0;
G(:,:,3)=0;
B(:,:,1)=0;
B(:,:,2)=0;
subplot(221)
imshow(RGB)
title('Original Image')
subplot(222)
imshow(R)
title('Red Component')
subplot(223)
imshow(G)
```

```
title('Green Component')
subplot(224)
imshow(B)
title('Blue Component')
```

Output:

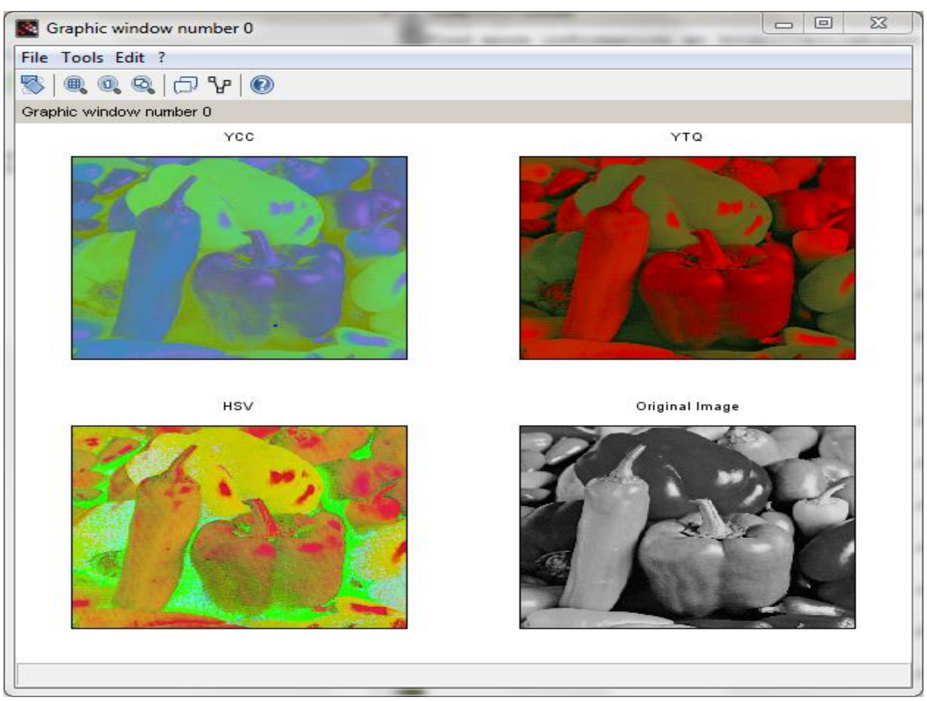


2) Program to convert RGB to different color format.

CODE:

```
RGB=imread(fullpath(getIPCVpath()+"images/peppers.png"));
G = rgb2gray(RGB)
YCC = rgb2ycbcr(RGB)
YIQ = rgb2ntsc(RGB)
HSV = rgb2hsv(RGB)
subplot(221)
imshow(YCC)
title('YCC')
subplot(222)
imshow(YIQ)
title('YTQ')
subplot(223)
imshow(HSV)
title('HSV')
subplot(224)
imshow(G)
title('Original Image')
```

OUTPUT:



<u>Conclusion:</u> We have successfully studied Binary Image Processing and Color Image Processing.