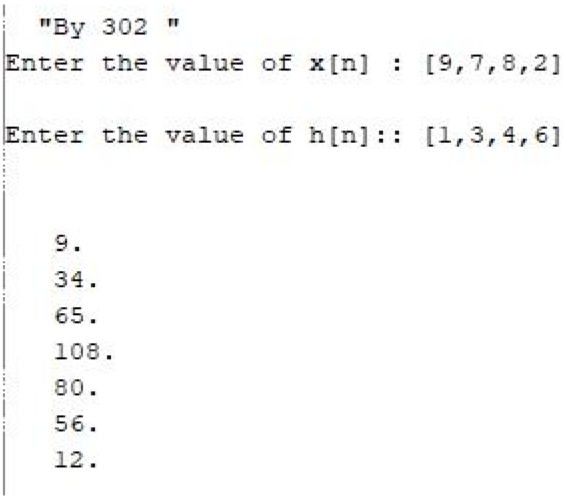
Practical 1 (Linear Convolution)

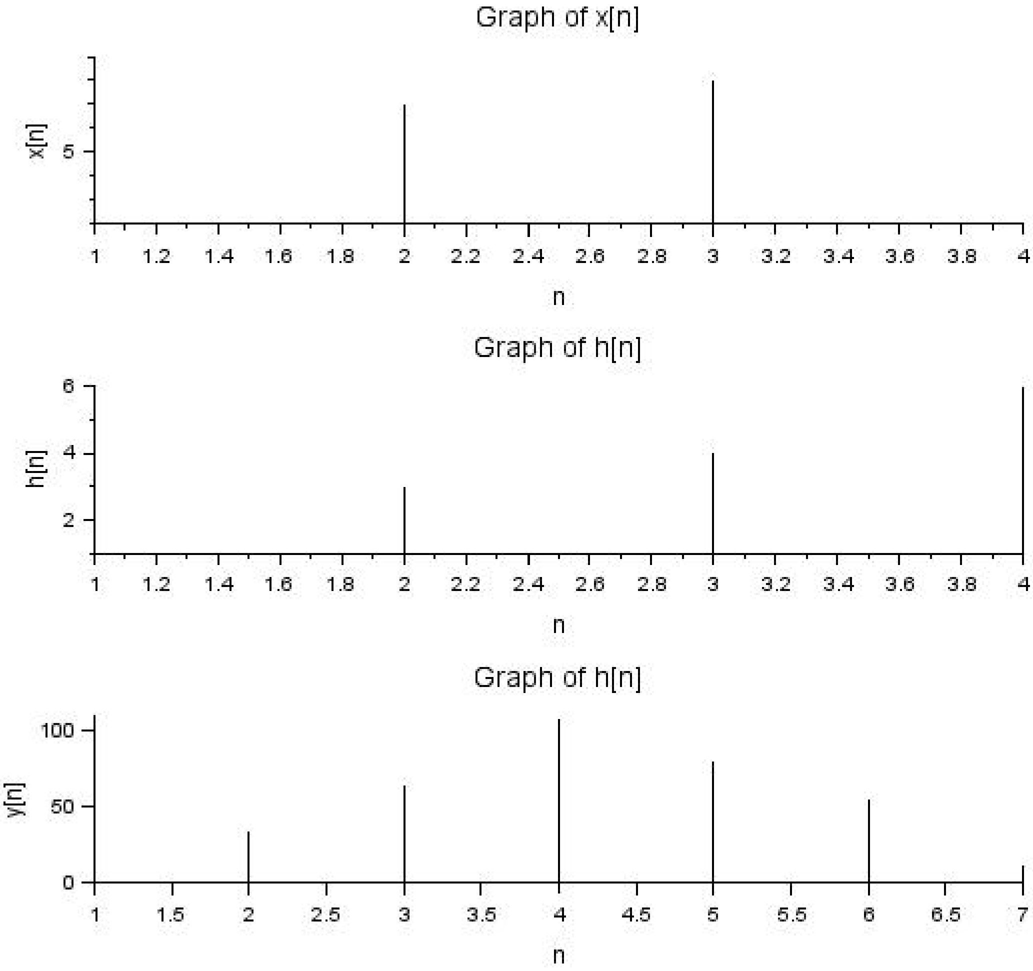
CODE:

```
//Linear Convolution
clc;
clf;
clear all;
disp("By 302 ");
x = input("Enter the value of x[n] : ");
h = input("Enter the value of h[n]:: ");
n1 = length(x);
n2 = length(h);
n = n1 + n2 -1;
subplot(3,1,1);
plot2d3(x);
xlabel("n");
ylabel("x[n]");
title("Graph of x[n] ");
subplot(3,1,2);
plot2d3(h);
xlabel("n");
ylabel("h[n]");
title("Graph of h[n] ");
x = [x, zeros(1,n-n1)];
h = [h, zeros(1,n-n2)];
for i = 1:n;
        conv_sum = 0;
        for j = 1:i;
                if(((i - j + 1)< = n1) & (j < = n2))
                        conv_sum = conv_sum + x(j) * h(i - j + 1);
                end
                y(i)= conv_sum;
        endend
        disp(y);
subplot(3,1,3);
plot2d3(y);
xlabel("n");
ylabel("y[n]");
title("Graph of h[n] ");
```

Output:



```
  "By 302 "
Enter the value of x[n] : [9,7,8,2]

Enter the value of h[n]:: [1,3,4,6]



   9.
   34.
   65.
   108.
   80.
   56.
   12.
```



Graph of x[n]

Graph of h[n]

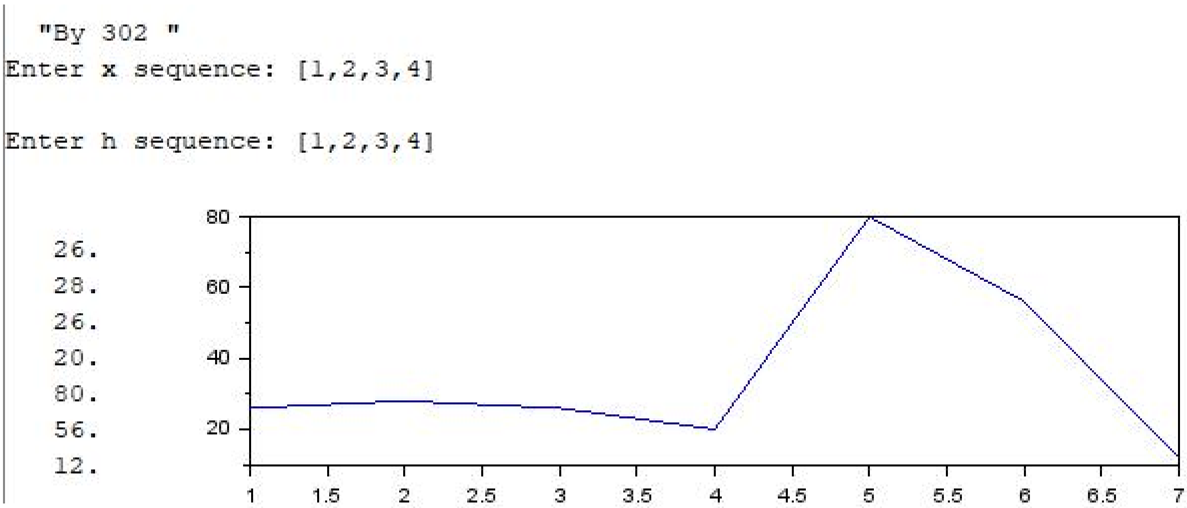
Graph of h[n]

Practical 2 (Circular Convolution)

Code:

```
//Circular Convolution
clc;
clf;
clear all;
disp("By 302 ");
g =input("Enter x sequence: ");
h =input("Enter h sequence: ");
n1 = length(g);
n2 = length(h);
n = max(n1,n2);
n3 = max(n1,n2);
n3 = n1-n2;
if(n3 > 0)
        h = [h, zeros(1,n3)];
else
        g = [g, zeros(1, -n3)];
end
for p = 1:n
        y (p) = 0;
        for q = 1:n
                j=p-q+1;
                if(j<=0)
                        j= n + j;
                end
                y(p) = [y(p) + g(q) * h(j)];
        end
end
disp(y);
subplot(3,1,1);
plot(y);
```

Output:



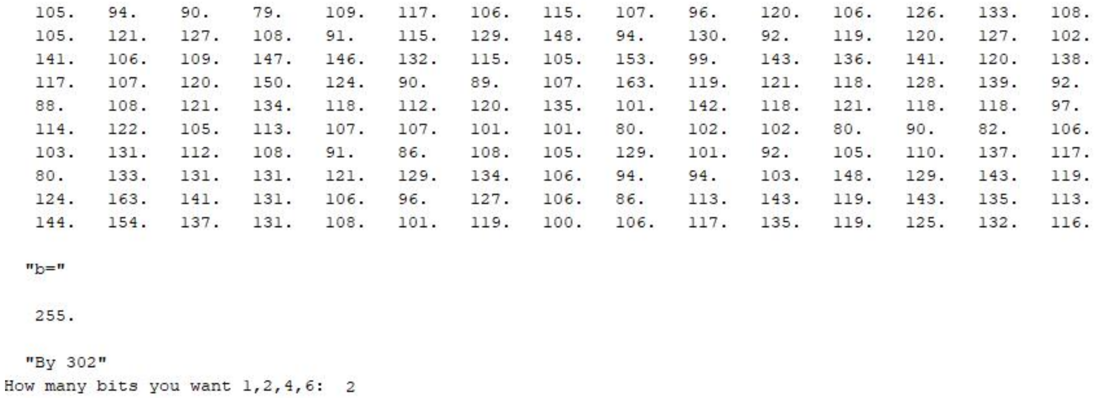
```
  "By 302 "
Enter x sequence:  [1,2,3,4]

Enter h sequence:  [1,2,3,4]

   26.
   28.
   26.
   20.
   80.
   56.
   12.
```

Practical 3 (Image Quantization)

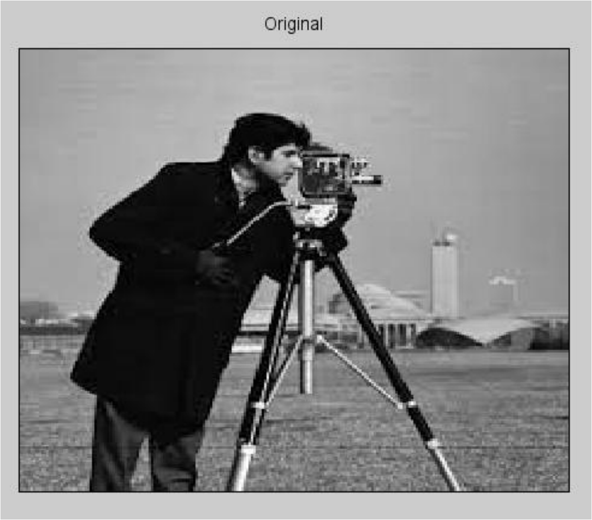
Code:

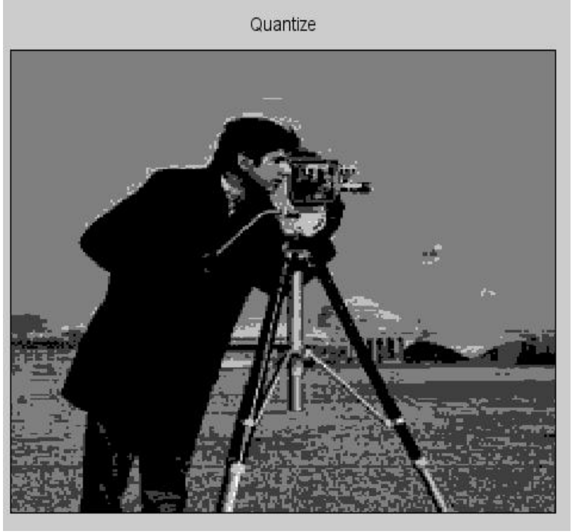
```
// Image Quantization
clc;
clear all;
I=imread("F:\cameraman.jpeg");
I=double(I);
disp("i===",I);
//disp(I);
//I=I+1
b=max(I)
disp("b=",b);
a=input("How many bits you want 1,2,4,6:")
c=b/(2*a);
f=floor(I/c);
disp("f=",f)
f1=(f*255)/max(f);
figure(1)
imshow(uint8(I))
figure(2)
imshow(uint8(f1))
```

Output:



```
   105.   94.    90.    79.    109.   117.   106.   115.   107.   96.    120.   106.   126.   133.   108.
   105.   121.   127.   108.   91.    115.   129.   148.   94.    130.   92.    119.   120.   127.   102.
   141.   106.   109.   147.   146.   132.   115.   105.   153.   99.    143.   136.   141.   120.   138.
   117.   107.   120.   150.   124.   90.    89.    107.   163.   119.   121.   118.   128.   139.   92.
   88.    108.   121.   134.   118.   112.   120.   135.   101.   142.   118.   121.   118.   118.   97.
   114.   122.   105.   113.   107.   107.   101.   101.   80.    102.   102.   80.    90.    82.    106.
   103.   131.   112.   108.   91.    86.    108.   105.   129.   101.   92.    105.   110.   137.   117.
   80.    133.   131.   131.   121.   129.   134.   106.   94.    94.    103.   148.   129.   143.   119.
   124.   163.   141.   131.   106.   96.    127.   106.   86.    113.   143.   119.   143.   135.   113.
   144.   154.   137.   131.   108.   101.   119.   100.   106.   117.   135.   119.   125.   132.   116.

  "b="

   255.

  "By 302"
How many bits you want 1,2,4,6:  2
```



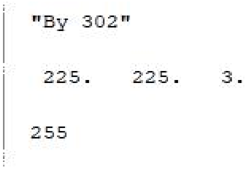


Practical 4 (Bit Resolution)

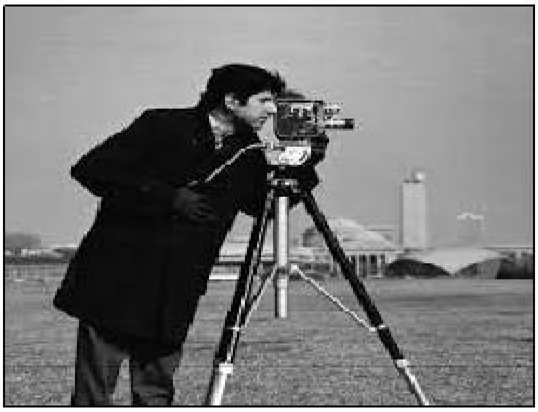
Code:

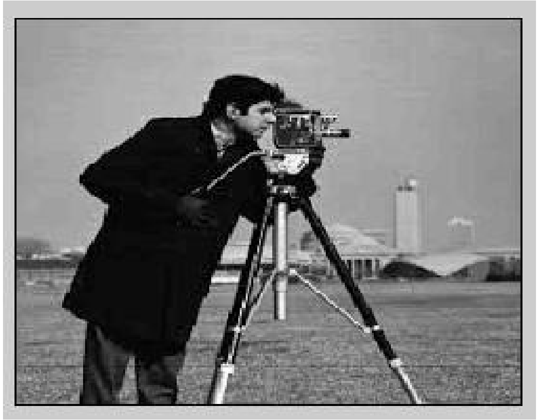
```
clc;
clf;
clear all;
close;
x=imread("F:\cameraman.jpeg");
disp("By 302");
imshow(x);
[r c s]=size(x);
disp([r c s]);
m=max(max(max(x)));
disp(m);
b=[2 3 4];
for i=1:length(b)
      d=2^b(i);
      z=round(x/d);
      figure
      imshow(z*d)
end
```

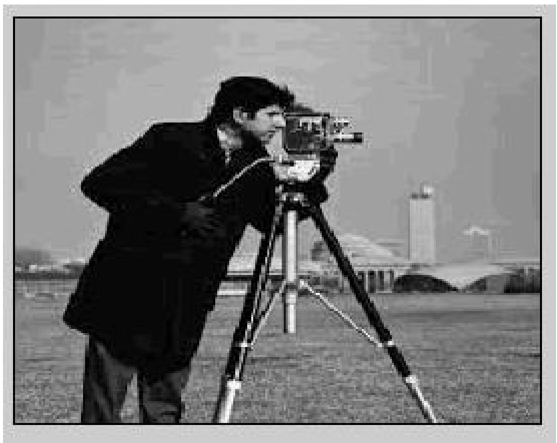
Output:

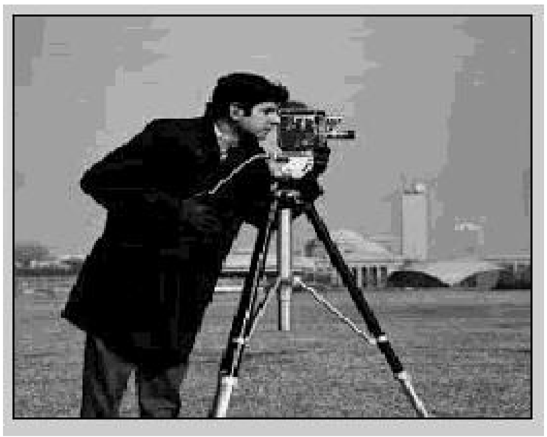


```
 "By 302"

  225.   225.   3.

 255
```







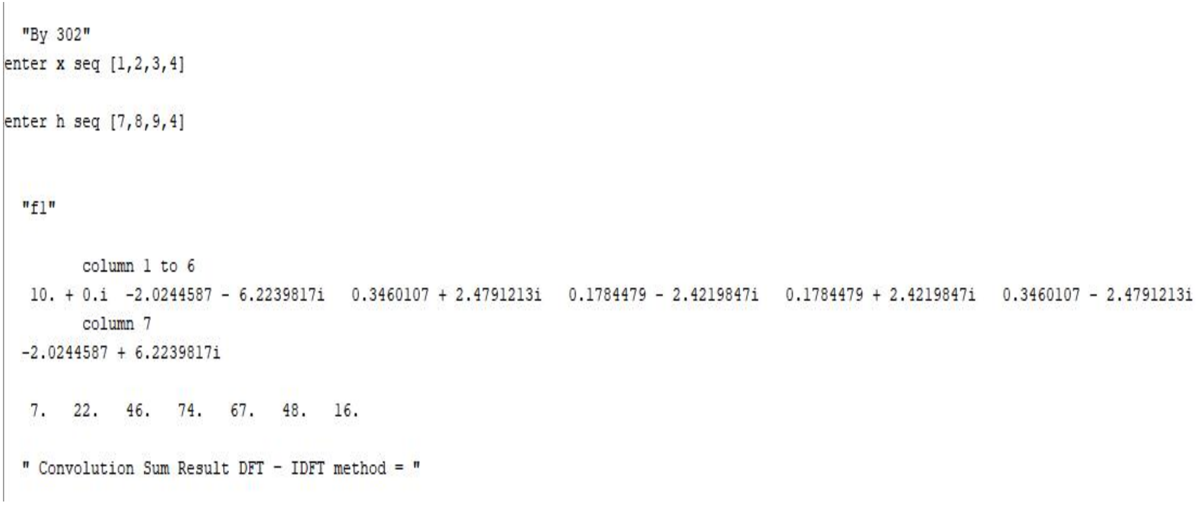


Practical 5 (DFT & IDFT)

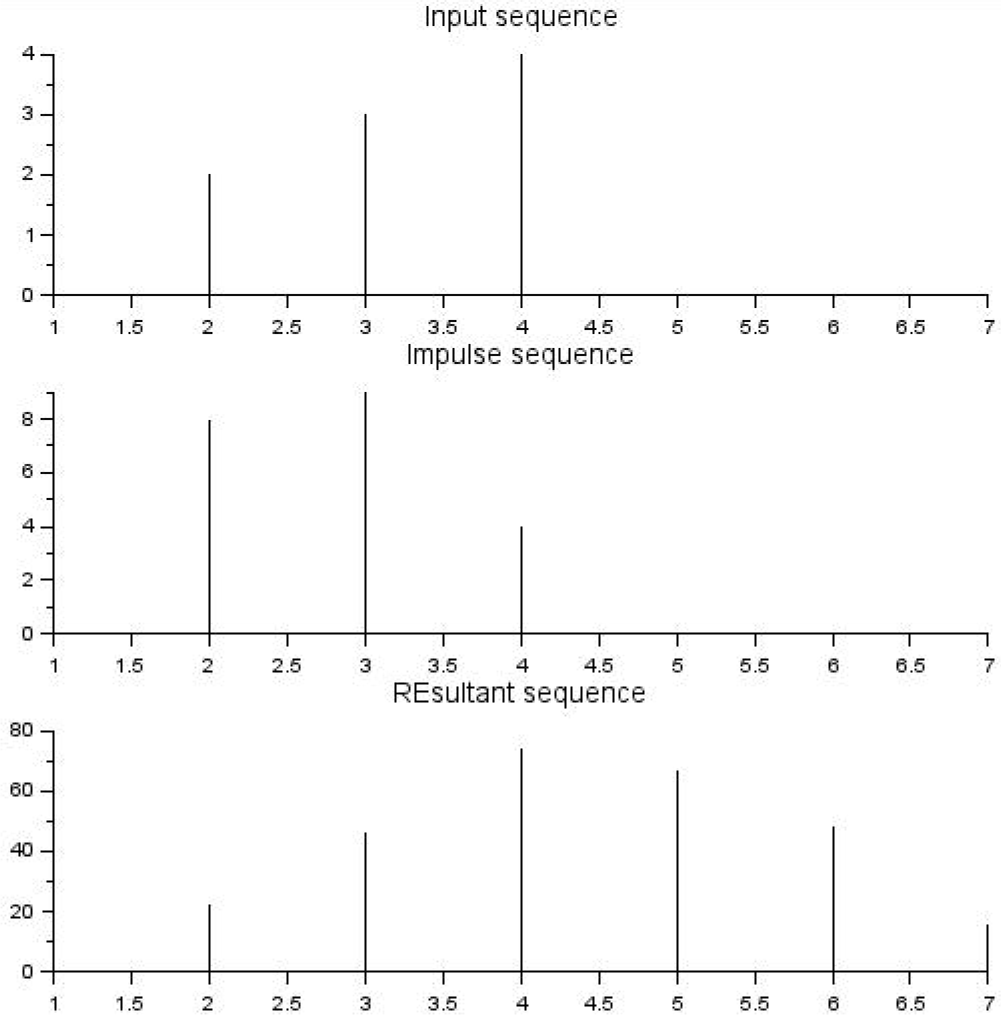
Code:

```
clc;
clear all;
close;
disp("By 302");
x = input ( " enter x seq " );
h = input ( "enter h seq " );
m = length ( x );
n = length ( h );
N = n +m -1;
x=[x,zeros(1,N-m)];
h=[h,zeros(1,N-n)];
f1 = fft( x )
disp("f1",f1)
f2 = fft( h )
f3 = f1 .* f2 ; // freq domain multiplication
f4 = ifft ( f3 )
disp ( f4 ," Convolution Sum Result DFT − IDFT method = ");// f4 = real (f4)
subplot (3 ,1 ,1);
plot2d3 ( x );
xtitle ( " Input sequence" );
subplot (3 ,1 ,2);
plot2d3 (h );
xtitle ( " Impulse sequence" );
subplot (3 ,1 ,3);
plot2d3 ( f4 );
xtitle ( " Resultant sequence ");// Result
```

Output:



```
  "By 302"
enter x seq [1,2,3,4]

enter h seq [7,8,9,4]


  "f1"


        column 1 to 6
   10. + 0.i  -2.0244587 - 6.2239817i   0.3460107 + 2.4791213i   0.1784479 - 2.4219847i   0.1784479 + 2.4219847i   0.3460107 - 2.4791213i
        column 7
  -2.0244587 + 6.2239817i


   7.   22.   46.   74.   67.   48.   16.

  " Convolution Sum Result DFT - IDFT method = "
```

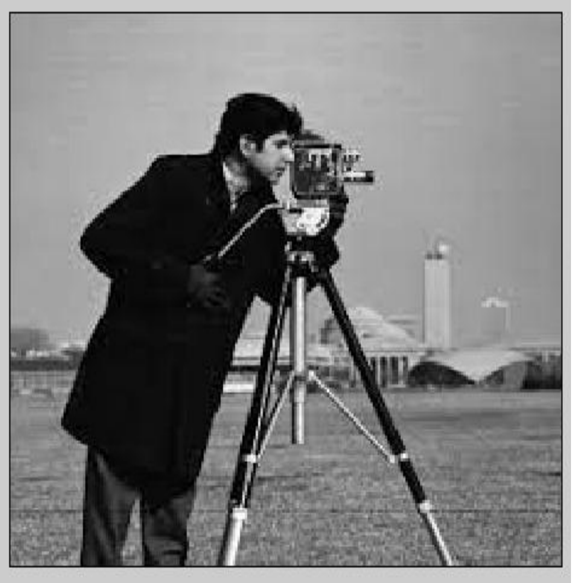


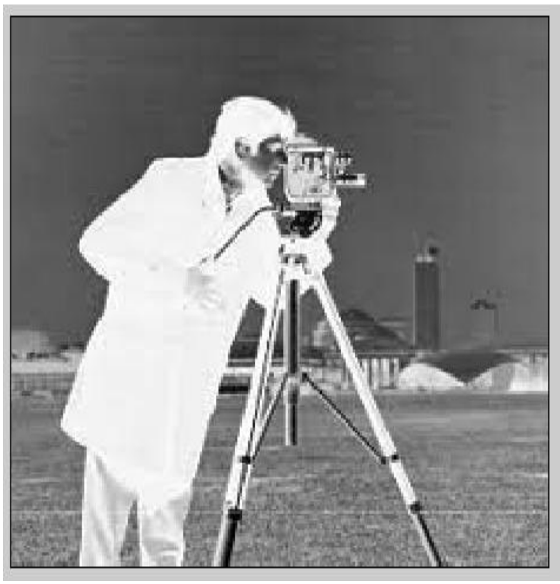
Practical 6

Image Negative Code:

```
clc;
original=imread("f:\dip\cameraman.jpeg");
imgdouble=double(original);//For 8 bit image
c=255;
negative=c-original;
figure(1)
imshow(original);
figure(2)
imshow(negative);
```

Output:

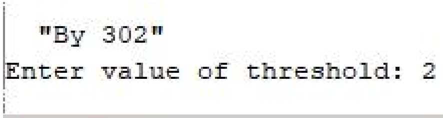




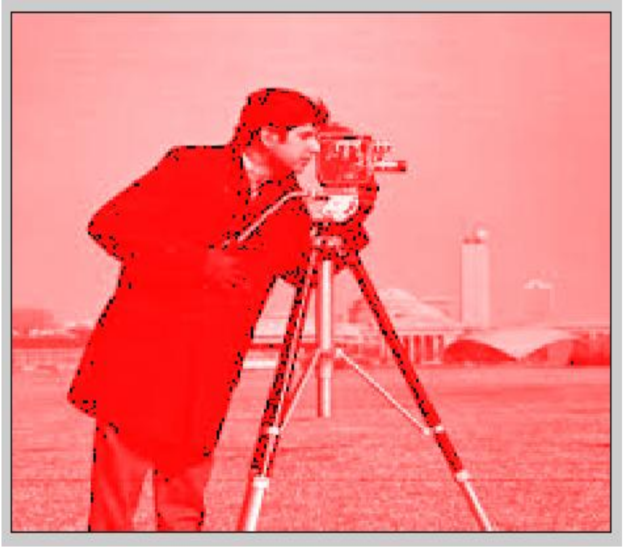
Threshold Code:

```
.clc;
original=imread("f:\dip\cameraman.jpeg");
dup=original;
[row column]=size(dup);
disp("By 302");
thresh=input("Enter value of threshold: ");
for i=1:row
        for j=1:column
                if(original(i,j)< thresh)
                        dup(i,j)=0;
                else
                        dup(i,j)=255;
                end
          End
End
figure(1),imshow(original);
figure(1),imshow(dup);
```

Output:



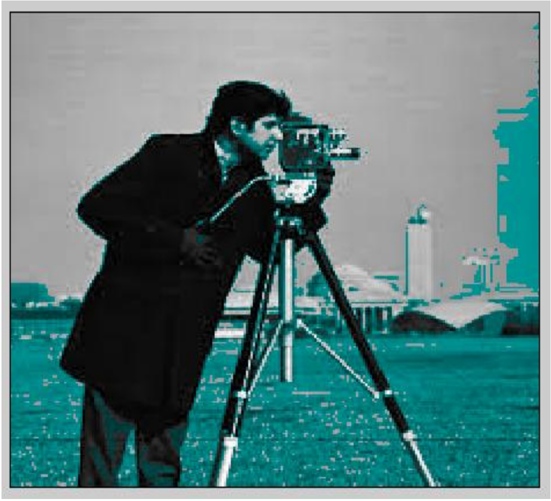
```
  "By 302"
Enter value of threshold: 2
```



Grey level slicing without background Code:

```
clc;
original=imread("f:\dip\cameraman.jpeg");
doub=double(original);
[row column]=size(doub);
for i=1:1:row
      for j=1:1:column
            if((doub(i,j)> 50)) && (doub(i,j)<150)
                  doub(i,j)=0;
            endendend
figure(1), imshow(original);
figure(2),imshow(uint8(doub));
```

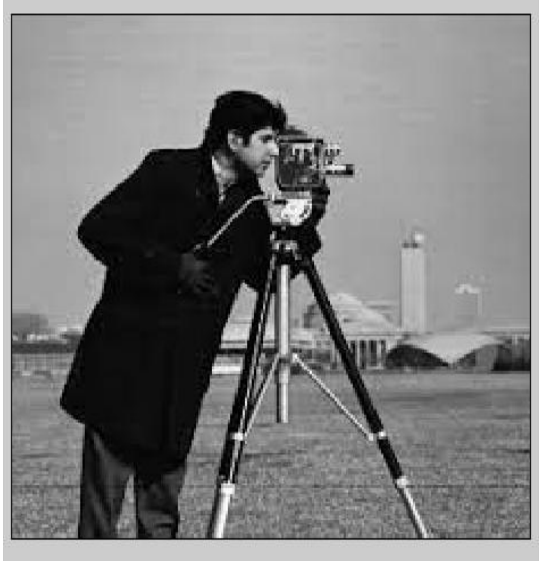
Output:



Grey level slicing with background Code:

```
clc;
original=imread("f:\dip\cameraman.jpeg");
doub=double(original);
[row column]=size(doub);
for i=1:1:row
    for j=1:1:column
        if((doub(i,j)> 50)) && (doub(i,j)<150)
            doub(i,j)=original(i,j);
        end
     end
end
figure(1), imshow(original);
figure(2),imshow(uint8(doub));
```

Output:

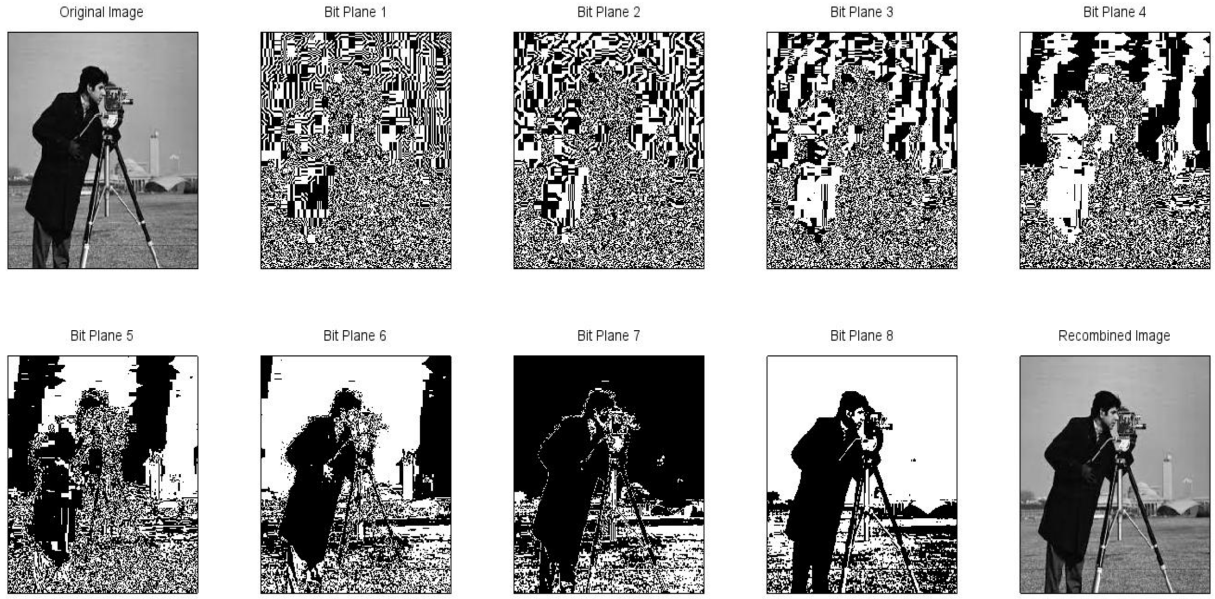


Bit Plane Slicing Code:

```
clc;
//reading image's pixel in c
c=imread("f:\dip\abc.jpeg");
//storing image information in cd
cd = double(c);
c1 = modulo(cd, 2);
c2 = modulo(floor(cd/2), 2);
c3 = modulo(floor(cd/4), 2);
c4 = modulo(floor(cd/8), 2);
c5 = modulo(floor(cd/16), 2);
c6 = modulo(floor(cd/32), 2);
c7 = modulo(floor(cd/64), 2);
c8 = modulo(floor(cd/128), 2);
//combining image again to form equivalent to original grayscale image
cc = (2 * (2 * (2 * (2 * (2 * (2 * (2 * c8 + c7) + c6) + c5) + c4) + c3) + c2) + c1);
//plotting original image in first subplot
subplot(2, 5, 1);
imshow(c);
```

```
title('Original Image');
//plotting binary image having extracted bit from //1st to 8th// in subplot from 2nd to
9th
subplot(2, 5, 2);
imshow(c1);
title('Bit Plane 1');
subplot(2, 5, 3);
imshow(c2);
title('Bit Plane 2');
subplot(2, 5, 4);
imshow(c3);
title('Bit Plane 3');
subplot(2, 5, 5);
imshow(c4);
title('Bit Plane 4');
subplot(2, 5, 6);
imshow(c5);
title('Bit Plane 5');
subplot(2, 5, 7);
imshow(c6);
title('Bit Plane 6');
subplot(2, 5, 8);
imshow(c7);
title('Bit Plane 7');
subplot(2, 5, 9);
imshow(c8);
title('Bit Plane 8');
//plotting recombined image in 10th subplot
subplot(2, 5, 10);
imshow(uint8(cc));
title('Recombined Image');
```

Output:

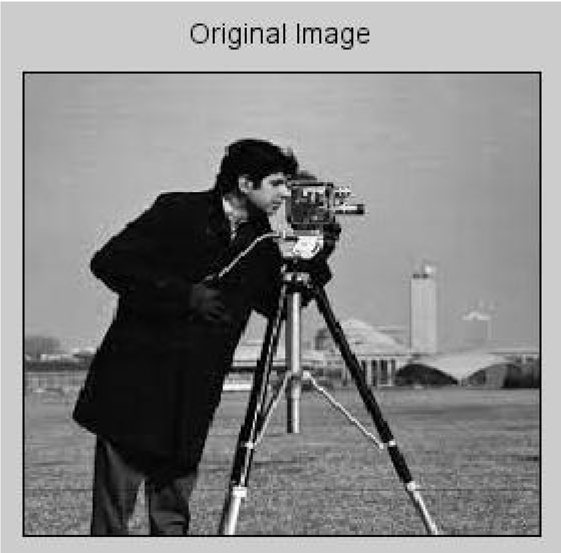


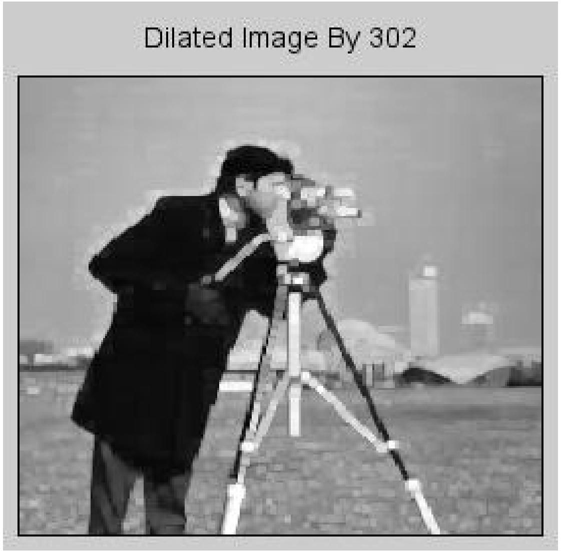
Practical 7 (Dil, Erd, Opn, Cls)

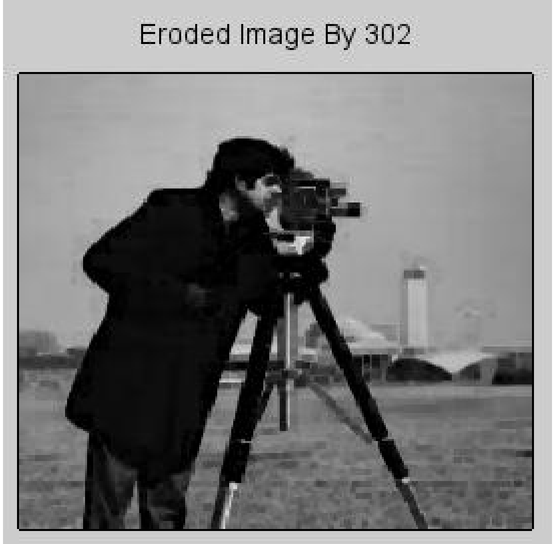
Code:

```
original=imread("f:\dip\cameraman.jpeg")
figure(1); title("Original Image") imshow(original);
//Specifing Structing Element as "Rectangle"
se=imcreatese("rect",3,3);
//dilation
dilate=imdilate(original,se);
figure(2); title("Dilated Image By 302") imshow(dilate);
//Erosion
erode=imerode(original,se);
figure(3); title("Eroded Image By 302") imshow(erode);
//Opening
afteropen=imopen(original,se);
figure(4); title("Opened Image By 302") imshow(afteropen);
//Closing
afterclose=imclose(original,se);
figure(5); title("Closed Image By 302") imshow(afterclose);
//Trying Different Structuring Element -> currently support 'rect', 'ellipse'
and 'cross'//Opening with SE -> Ellipse
S=imread("f:\dip\cameraman.jpeg");
se3 = imcreatese('ellipse',9,9);
S2 = imopen(S,se3);
figure(7); title("Opening with SE -> Ellipse By 302") imshow(S2);
//Closing with SE -> Ellipse
se = imcreatese('ellipse',11,11);
S2 = imclose(S,se);
figure(6); title("Closing with SE -> Ellipse By 302") imshow(S2);
```

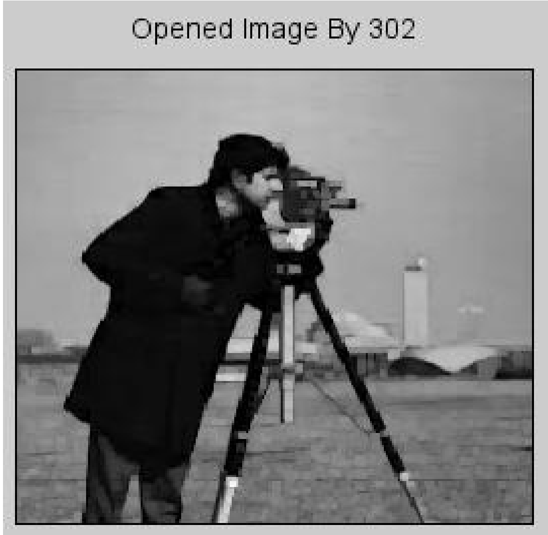
Output:


Original Image

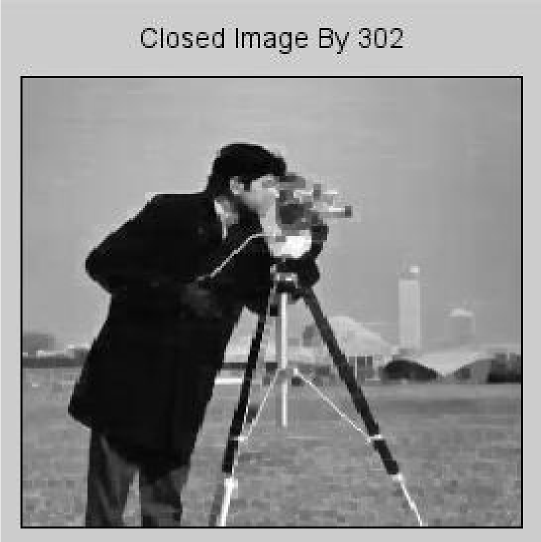

Dilated Image By 302



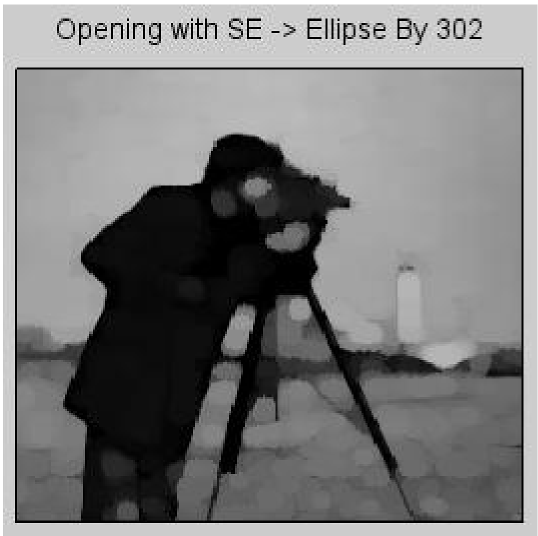
Eroded Image By 302



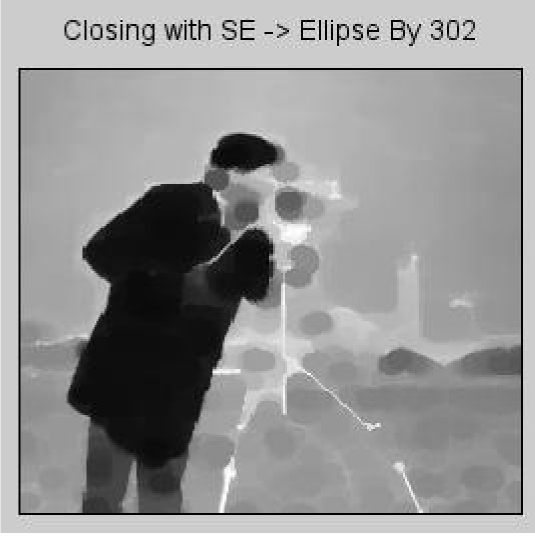
Opened Image By 302



Closed Image By 302



Opening with SE -> Ellipse By 302



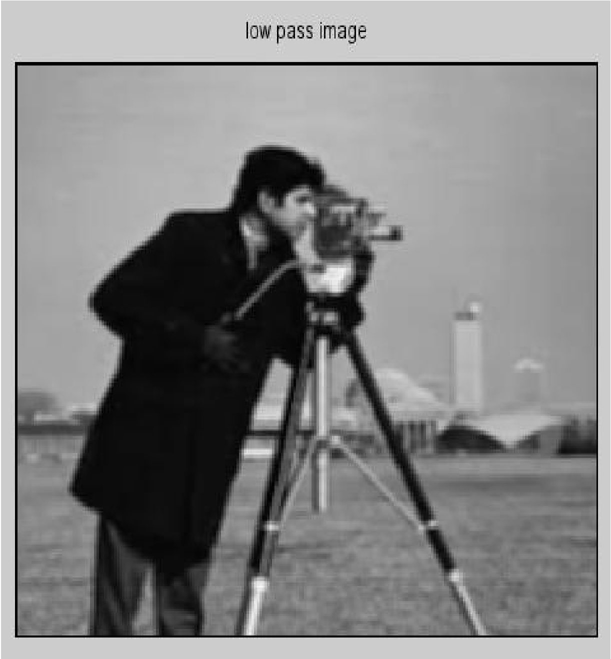
Closing with SE -> Ellipse By 302

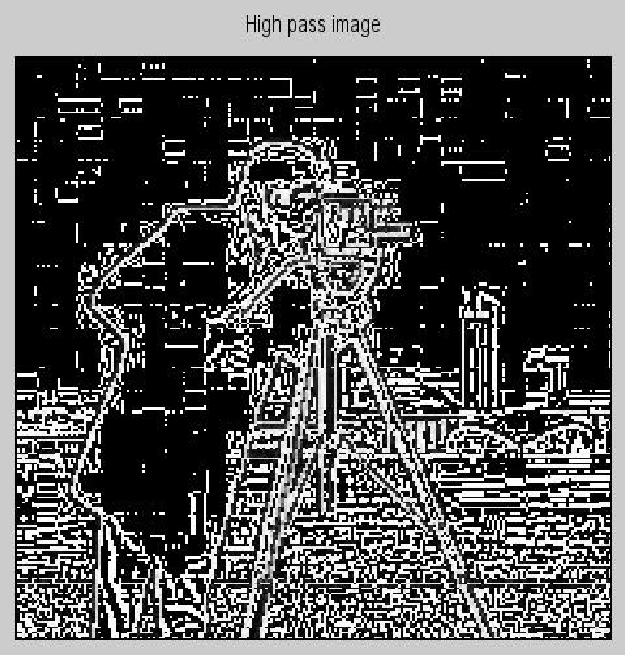
Practical 8 (LDF & HPF)

Code:

```
//Low Pass Filter
clc;
a1=imread("f:\dip\cameraman.jpeg");
a=double(a1);
[m,n]=size(a);
w=[1 1 1;1 1 1;1 1 1];
for i=2:m-1
    for j=2:n-1
         b(i,j)=[w(1)*a(i-1,j+1)+w(2)*a(i,j+1)+w(3)*a(i+1,j+1)+w(4)*a(i-1,j)
         +w(5)*a(i,j)+w(6)*a(i+1,j)+w(7)*a(i-1,j-1)+w(8)*a(i,j-1)+w(9)*a(i+
         1,j-1)]/9;
     endend
c=uint8(b);
figure(1);
imshow(c);
title("low pass image")

//High Pass Filter
clc;
a1=imread("f:\dip\cameraman.jpeg");
a=double(a1);
[m,n]=size(a);
w=[-1 -1 -1;-1 8 -1;-1 -1 -1];
for i=2:m-1
     for j=2:n-1
         H(i,j)=[w(1)*a(i-1,j+1)+w(2)*a(i,j+1)+w(3)*a(i+1,j+1)+w(4)*a(i-1,j)
         +w(5)*a(i,j)+w(6)*a(i+1,j)+w(7)*a(i-1,j-1)+w(8)*a(i,j-1)+w(9)*a(i+
         1,j-1)]/9;
     endend
D=uint8(H);
figure(2);
imshow(D);
title("High pass image")
figure(3);
imshow(a1);
title("Original image")
```

Output:




High pass image