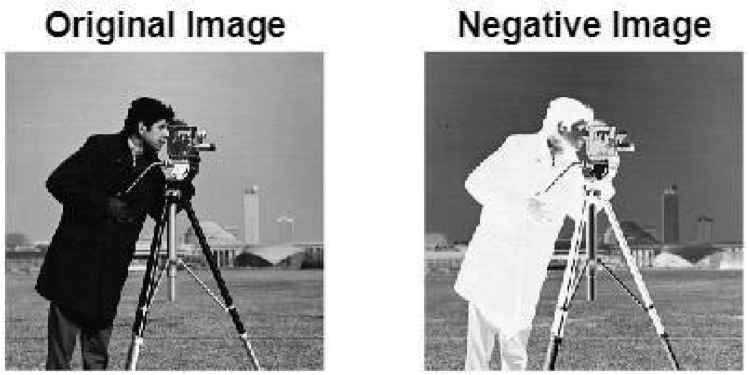
```
% Expt. No.2
% Aim: Perform the following basic operations on image:
% a. Obtain Negative image
% b. Obtain Flip image

clc;
clear all;
close all;
I = imread('cameraman.tif');
I1 = 255-I;
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I1), title('Negative Image');
```

Output:



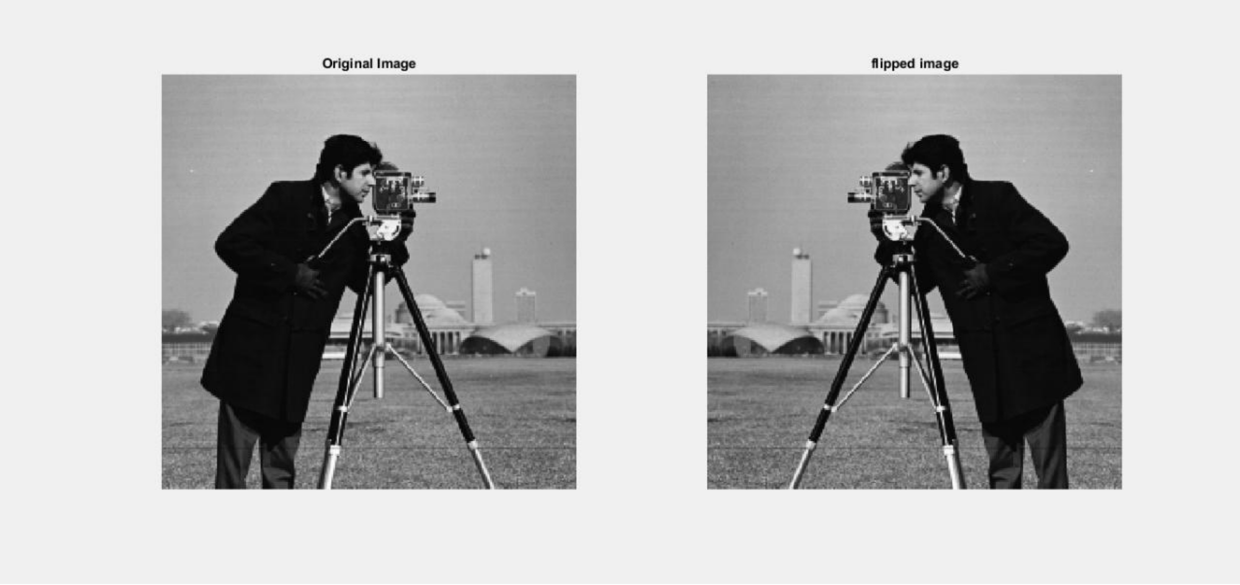
% Expt. No. 2

% Aim: Perform the following basic operations on image:

% b. Obtain Flip image

```
 clc;
clear all;
close all;


 org_img = imread('cameraman.tif');
 M = size(org_img,1);
 N = size(org_img,2);
 for  i = 1:M
     for j = 1:N
    n_img(i,j) = org_img(i,N-j+1)
     subplot(4,3,2), imshow( n_img(i,j));
     end
end
```

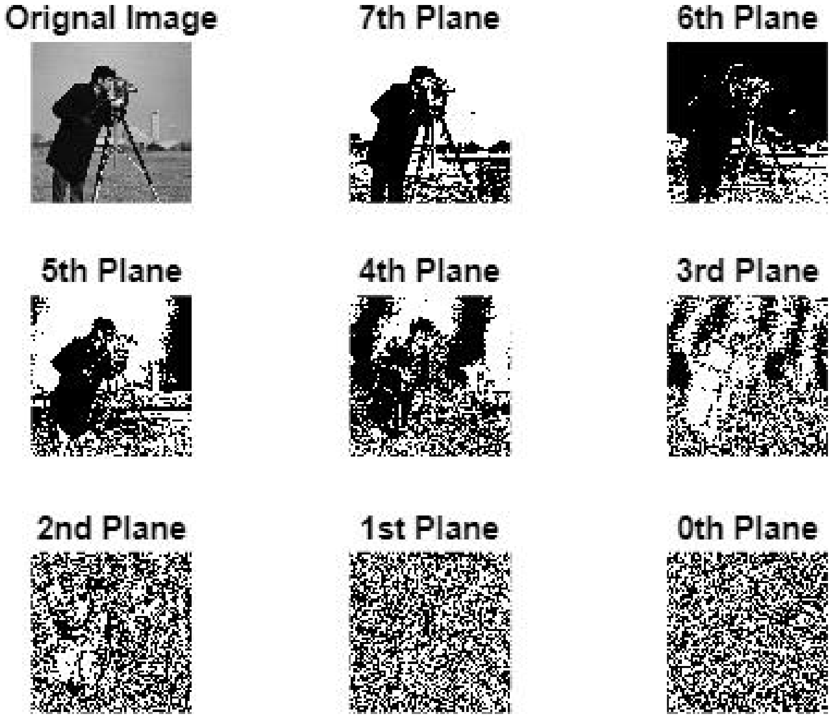
Output:



```
% Expt. No. 3

% Aim: Read an image and to extract 8 different planes  i.e 'bit plane slicing'

clc;
clear all;

close all;


f = imread('cameraman.tif');

b7 = (zeros(256));

b6 = (zeros(256));

b5 = (zeros(256));

b4 = (zeros(256));

b3 = (zeros(256));

b2 = (zeros(256));

b1 = (zeros(256));

b0 = (zeros(256));


for i = 1:256

for j = 1:256

y = fliplr(dec2bin(f(i,j),8));

b0(i,j) = bin2dec(y(1));

b1(i,j) = bin2dec(y(2));

b2(i,j) = bin2dec(y(3));

b3(i,j) = bin2dec(y(4));
b4(i,j) = bin2dec(y(5));

b5(i,j) = bin2dec(y(6));

b6(i,j) = bin2dec(y(7));

b7(i,j) = bin2dec(y(8));

end

end
```

```
subplot(3,3,1); imshow(f); title('Orignal Image');
subplot(3,3,2); imshow(b7); title('7th Plane');
subplot(3,3,3); imshow(b6); title('6th Plane');
subplot(3,3,4); imshow(b5); title('5th Plane');
subplot(3,3,5); imshow(b4); title('4th Plane');
subplot(3,3,6); imshow(b3); title('3rd Plane');
subplot(3,3,7); imshow(b2); title('2nd Plane');
subplot(3,3,8); imshow(b1); title('1st Plane');
subplot(3,3,9); imshow(b0); title('0th Plane');
```

Output:

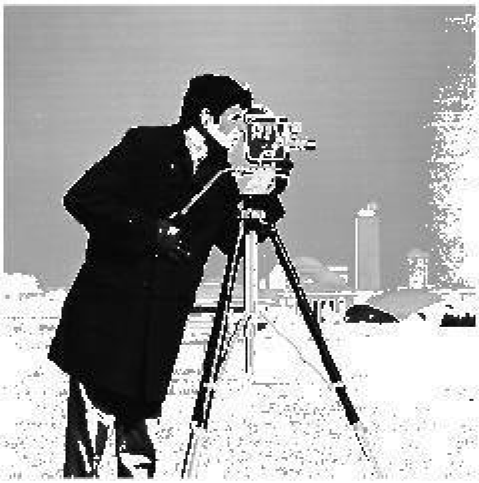


% Expt. No.3

% Grey level slicing with background

```
 clc;
clear all;
close all;


p = imread('cameraman.tif');
z = p;
   [m,n] = size(p); for
  i = 1:m
      for j = 1:n if((z(i,j))>50)&&(z(i,j)<150)
             z(i,j) = 255;
          else
             z(i,j) = p(i,j); %condition for grey level slicing with background  end
      end
      end
  figure(1);
  imshow(p), title('Orignal Image')
  figure(2);
  imshow(z), title('Grey Level Slicing With Background');
```

% Output:

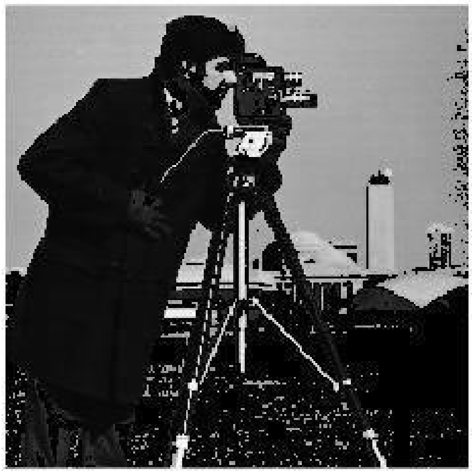


Grey Level Slicing With Background

```
% Expt. No.3
% Grey level slicing without background
 clc;
clear all;
close all;
p = imread('cameraman.tif');
z = p;
[m,n] = size(p);
for i = 1:m
    for j = 1:n if((z(i,j))>50)&&(z(i,j)<150)
          z(i,j) = 0;
       else
          z(i,j) = p(i,j); % Condition for grey level slicing with background
          end
    end
    end
  figure(1);
  imshow(p), title('Orignal Image') figure(2);
  imshow(z), title('Grey Level Slicing Without Background')
```

Output:

Orignal Image



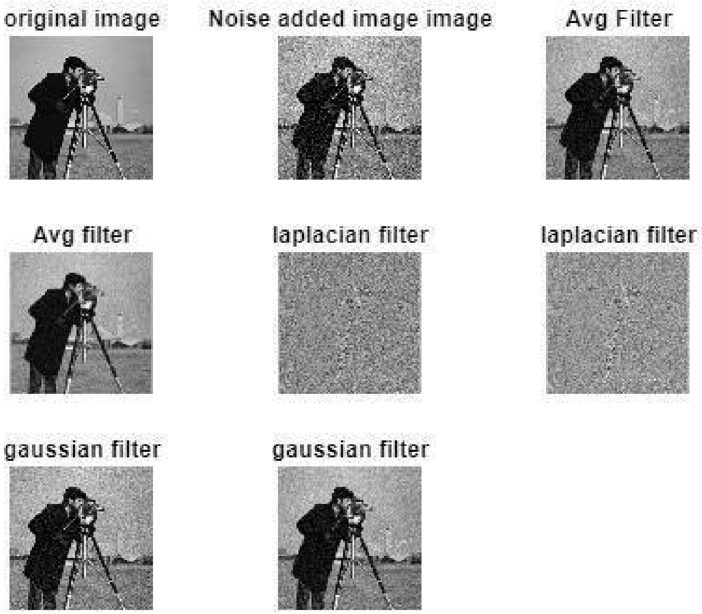
Grey Level Slicing Without Background

% Expt. No.: 4

% Aim/Title: Write a program to implement image filtering in spatial domain (Using Readymade)

```
clc;
clear all;
close all;

J = imread('cameraman.tif');
I = imnoise(J,'gaussian');
I = double(I);
% Readymade function
m1 = fspecial('average',2);
m2 = fspecial('average',3);
m3 = fspecial('laplacian',0.2);
m4 = fspecial('laplacian',0.6);
m5 = fspecial('gaussian',3);
m6 = fspecial('gaussian',2);
% Convolution of filters
o4 = conv2(I,m1,'same');
o5 = conv2(I,m2,'same');
o6 = conv2(I,m3,'same');
o7 = conv2(I,m4,'same');
o8 = conv2(I,m5,'same');
o9 = conv2(I,m6,'same');

figure
subplot(331), imshow(J), title('original image');

subplot(332), imshow(uint8(I)), title('Noise added image image');

subplot(333), imshow(o4,[]), title('Avg Filter');
subplot(334), imshow(o5,[]), title('Avg filter');
subplot(335), imshow(o6,[]), title('laplacian filter');
subplot(336), imshow(o7,[]), title('laplacian filter');
subplot(337), imshow(o8,[]), title('gaussian filter');
subplot(338), imshow(o9,[]), title('gaussian filter');
```

Output:



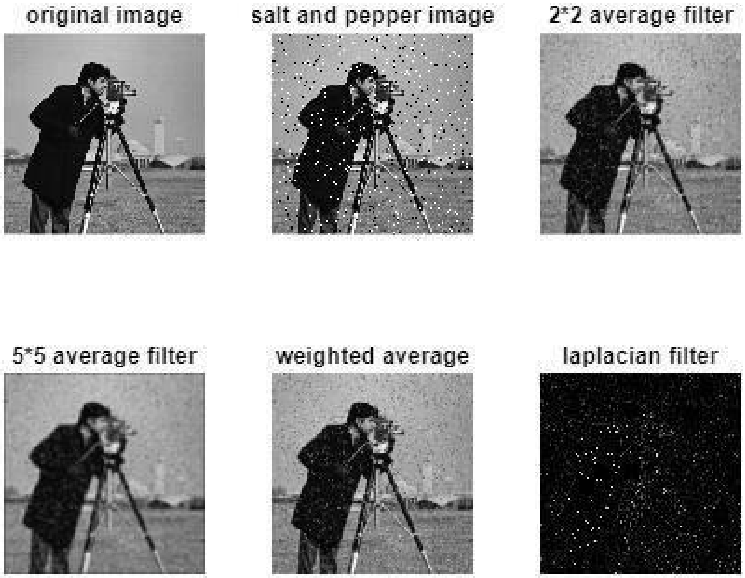
% Expt. No.: 4

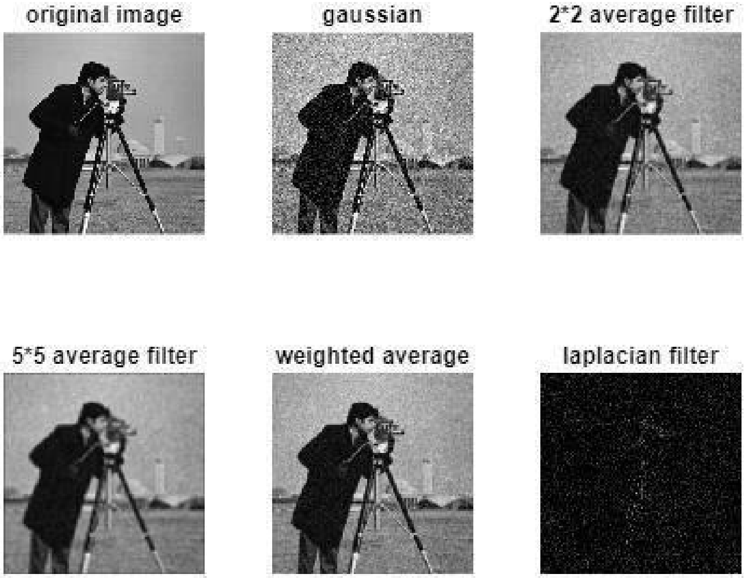
% Aim/Title: Write a program to implement image filtering in spatial domain (Using User defined)

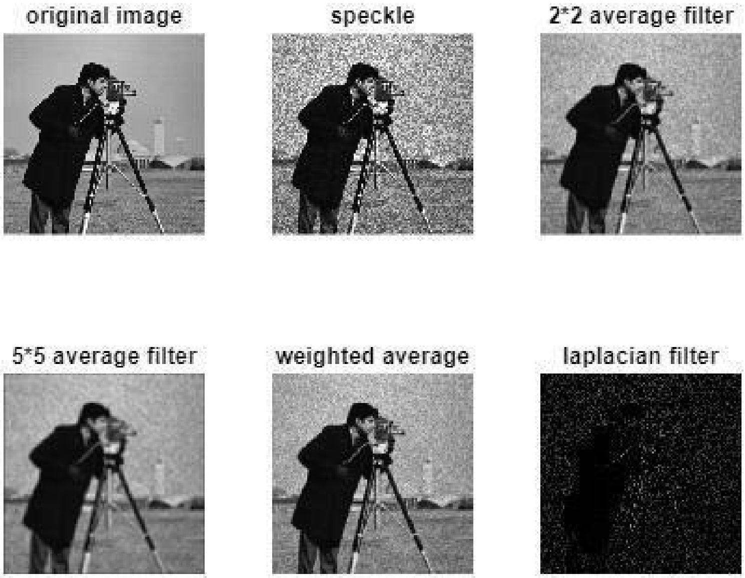
```
clc;
clear all;
close all;

a = imread('cameraman.tif');
b = imnoise(a,'salt & pepper');
b = double(b);
c = imnoise(a,'gaussian');
c = double(c);
d = imnoise(a,'speckle');
d = double(d);
h1 = 1/9*ones(3,3); % 3*3 average
h2 = 1/25*ones(5,5); %  5*5 average
h3 = (1/16).*[1,2, 1; 2, 4, 2; 1, 2, 1]; %weighted average
h4 = (1/9).*[-1,-1,-1;-1,8,-1;-1,-1,-1]; %laplacian filter

b1 = conv2(b,h1,'same');
b2 = conv2(b,h2,'same');
b3 = conv2(b,h3,'same');
b4 = conv2(b,h4,'same');
c1 = conv2(c,h1,'same');
c2 = conv2(c,h2,'same');
c3 = conv2(c,h3,'same');
c4 = conv2(c,h4,'same');
d1 = conv2(d,h1,'same');
d2 = conv2(d,h2,'same');
d3 = conv2(d,h3,'same');
d4 = conv2(d,h4,'same');
```

```
% Salt and pepper figure;
subplot(231), imshow(a), title('original image');
subplot(232), imshow(uint8(b)), title('salt and pepper image');
subplot(233), imshow(uint8(b1)), title('2*2 average filter');
subplot(234), imshow(uint8(b2)), title('5*5 average filter');
subplot(235), imshow(uint8(b3)), title('weighted average');
subplot(236), imshow(uint8(b4)), title('laplacian filter');

% Gaussian
figure;
subplot(231), imshow(a), title('original image');
subplot(232), imshow(uint8(c)), title('gaussian');
subplot(233), imshow(uint8(c1)), title('2*2 average filter');
subplot(234), imshow(uint8(c2)), title('5*5 average filter');
subplot(235), imshow(uint8(c3)), title('weighted average');
subplot(236), imshow(uint8(c4)), title('laplacian filter');

% Speckle

figure;
subplot(231), imshow(a), title('original image');
subplot(232), imshow(uint8(d)), title('speckle');
subplot(233), imshow(uint8(d1)), title('2*2 average filter');
subplot(234), imshow(uint8(d2)), title('5*5 average filter');
subplot(235), imshow(uint8(d3)), title('weighted average');
subplot(236), imshow(uint8(d4)), title('laplacian filter');
```

Output:

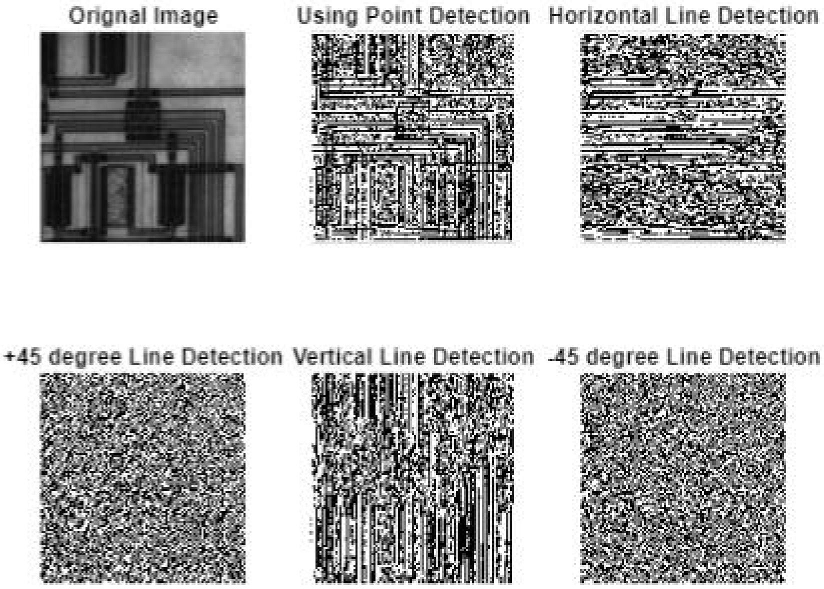

original image
salt and pepper image
2*2 average filter
5*5 average filter
weighted average
laplacian filter


original image
gaussian
2*2 average filter
5*5 average filter
weighted average
laplacian filter


original image
speckle
2*2 average filter
5*5 average filter
weighted average
laplacian filter

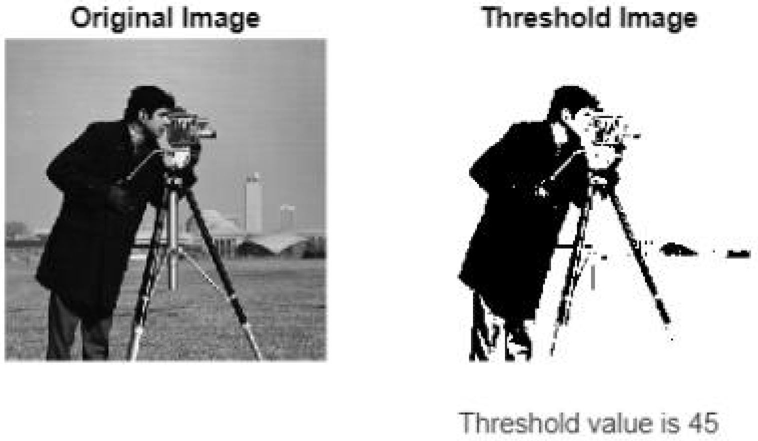
```
% TE Expt No.5
% Aim: Perform the following basic operations on image:-
% a. Point Detection, b. Line Detection, c. Edge Detection, d. Thresholding

clc;
clear all;
close all;

% a. Point Detection
% I=imread('cameraman.tif');
I=imread('circuit.tif');
I = double(I);

I11 = [-1 -1 -1; -1 8 -1; -1 -1 -1];
I22 = conv2(I, I11);

subplot(2,3,1); imshow(uint8(I));title('Orignal Image');
subplot (2,3,2); imshow(I22), title('Using Point Detection');

% b. Line Detection
I1 = [-1 -1 -1;2 2 2; -1 -1 -1];
I2 = [-1 -1 2;-1 2 -1; 2 -1 -1];
I3 = [-1 2 -1;-1 2 -1; -1 2 -1];
I4 = [2 -1 -1;-1 2 -1; -1 -1 2];
I5 = conv2(I, I1);
I6 = conv2(I, I2);
I7 = conv2(I, I3);
I8 = conv2(I, I4);

subplot (2,3,3); imshow(I5), title('Horizontal Line Detection');
subplot (2,3,4); imshow(I6), title('+45 degree Line Detection');
subplot (2,3,5); imshow(I7), title('Vertical Line Detection');
subplot (2,3,6); imshow(I8), title('-45 degree Line Detection');
```

Output:-


Orignal Image
Using Point Detection
Horizontal Line Detection
+45 degree Line Detection
Vertical Line Detection
-45 degree Line Detection

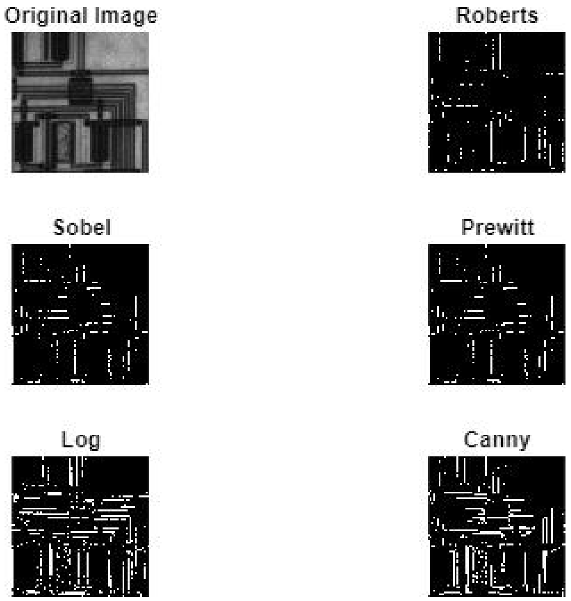
```
% TE Expt No.5
% Aim:Perform the following operations on image :
% a)Point detection   % b)Line detection
% c)Age detection    % D)Thresholding
%D)Thresholding
clc;
clear all;
close all;

a=imread('cameraman.tif');
[m n]=size(a);
t=input('Enter the Threshold parameter:');
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
subplot(1,2,1),imshow(a),title('Original Image');
subplot(1,2,2),imshow(b),title('Threshold Image');
xlabel(sprintf('Threshold value is %g',t));
```

```
% OUTPUT
% Enter the Threshold parameter:45
```



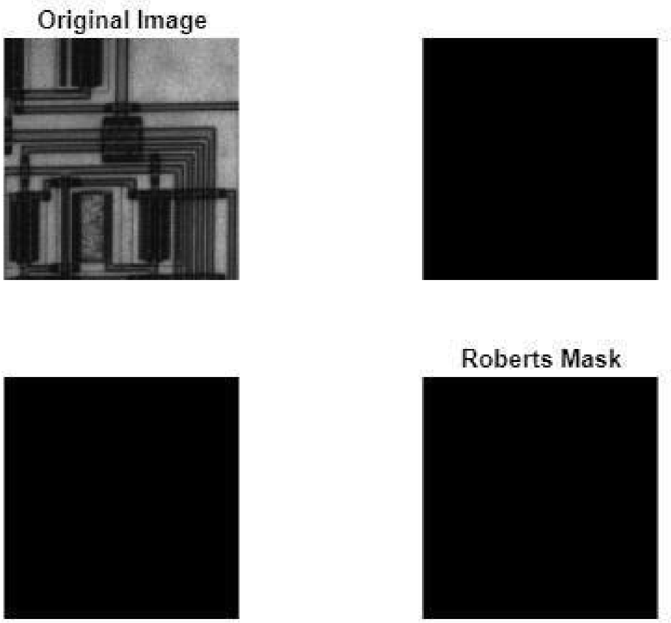
```
% Expt. No. 6
% Title/Aim: Implement and study the effect of Different Mask  (Sobel, Prewitt  and
Roberts)  (Readymade Function)

clc;
clear all;
close all;


% Using Readymade inbuilt function

a = imread('circuit.tif');
b = edge(a, 'roberts');
c = edge(a, 'Sobel');
d = edge(a, 'Prewitt');
e = edge(a, 'log');
f = edge(a, 'canny');
subplot(3,2,1); imshow(a),  title('Original Image');

subplot(3,2,2); imshow(b), title('Roberts');
subplot(3,2,3); imshow(c), title('Sobel');
subplot(3,2,4); imshow(d), title('Prewitt');
subplot(3,2,5); imshow(e), title('Log');
subplot(3,2,6); imshow(f), title('Canny');
```

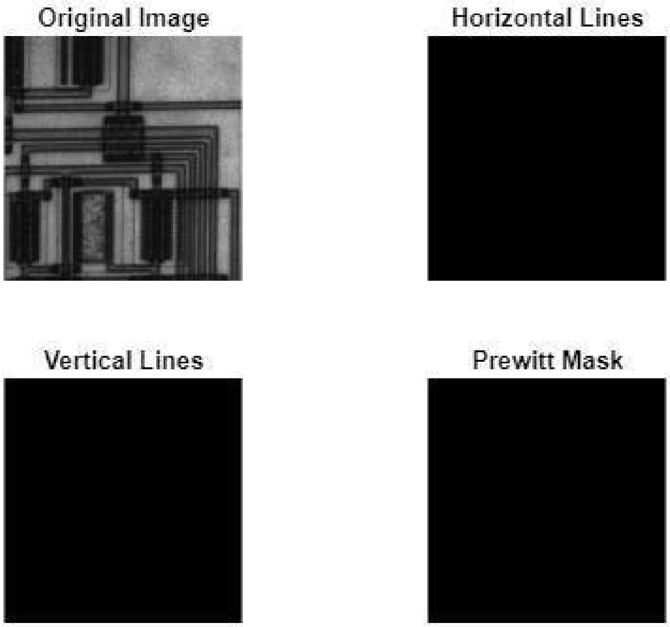
Output:



```
% Expt. No. 6
% Aim: Implement and study the effect of Different Mask  (Sobel, Prewitt and  Roberts)

clc;
clear all;
close all;
% Using userdefined function
% Mask for Roberts
a1 = imread('circuit.tif');

a1 = im2double(a1);
h5  = [1 0;0 -1];
M = conv2(a1,h5,'same');
h6 = [0 1;-1 0];
N = conv2(a1,h6,'same');

O = imadd(M,N);
figure;
subplot(221),  imshow(a1), title('Original Image');

subplot(222),  imshow(uint8(M));
subplot(223),  imshow(uint8(N));
subplot(224),  imshow(uint8(O)), title('Roberts Mask');

% Mask for Prewitt
h1 = [1 1 1;0 0 0; -1 -1 -1];
g  = conv2(a1,h1,'same');
h2 = [-1 0 1;-1 0 1;-1 0 1];
h = conv2(a1,h2,'same');

I = imadd(g,h);

figure;
subplot(221),  imshow(a1), title('Original Image');

subplot(222),  imshow(uint8(g)), title('Horizontal Lines');  subplot(223),
imshow(uint8(h)), title('Vertical Lines');

subplot(224),  imshow(uint8(I)), title('Prewitt Mask');

% Mask for Sobel
h3 = [-1 0 1;-2 0 2;-1 0 1];
J  = conv2(a1,h3,'same');
h4  =  [1 2 1;0 0 0;-1 -2 -1];
K = conv2(a1,h4,'same');

L = imadd(J,K);
figure;
subplot(221), imshow(a1), title('Original Image');

subplot(222),  imshow(uint8(J)), title('Vertical Lines');

subplot(223),  imshow(uint8(K)), title('Horizontal Lines');  subplot(224),
imshow(uint8(L)), title('Sobel Mask');
```

Output:



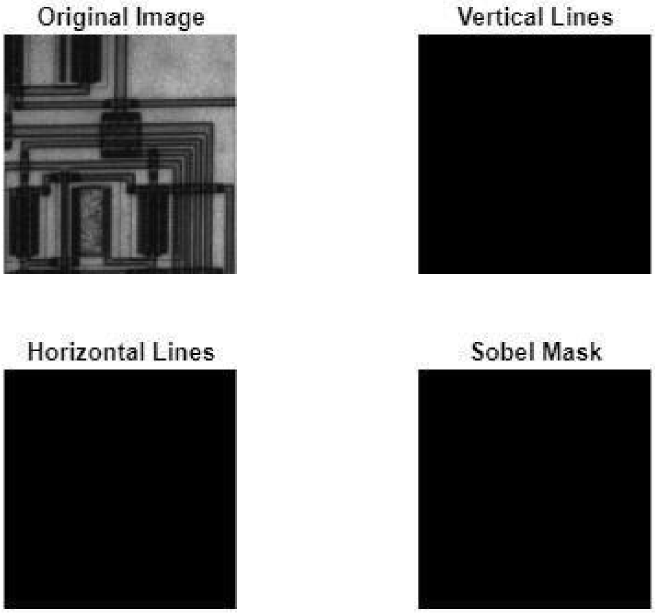


Original Image

Horizontal Lines

Vertical Lines

Prewitt Mask



Original Image

Vertical Lines

Horizontal Lines

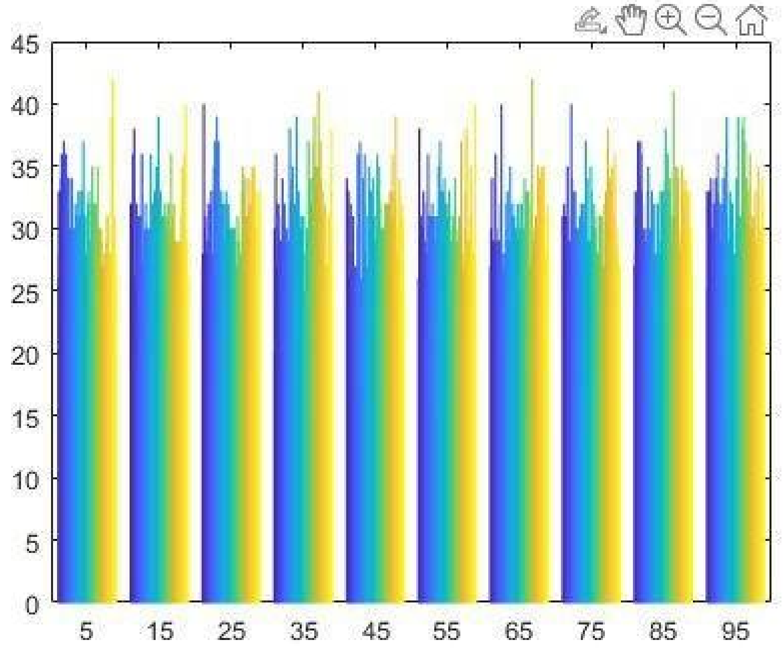
Sobel Mask

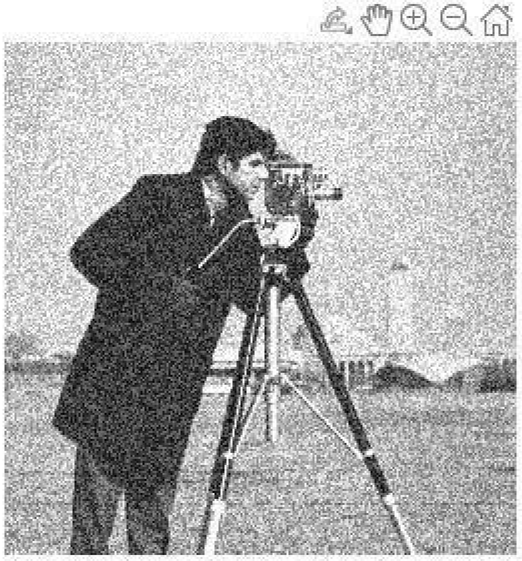
```
% Expt. No.7
% Implement various noise models and their histogram
clc;
clear    all;
close all;

a = 0;
b = 1;
f = imread('cameraman.tif');
f1 = double(f);
[r,c] = size(f);
I = input('What noise do you want to add? Uniform = 1; Rayleigh = 2; Expo = 3:');
if I == 1
    R = a+(b-a)*rand(r,c); %Uniform
    elseif I == 2
        R = a+(-b*log(1-rand(r,c))).^0.5;%Rayleigh
        elseif I == 3
            R = -log(1-rand(r,c)); %Exponential
end
        mmax  = max(max(R));

         mmin = min(min(R));

        const = 100/(mmax-mmin); for
        x = 1:1:r
            for y = 1:1:c
            noise(x,y) = const*(R(x,y)- mmin);
            end
```

```
end
noisy_image = f1 + noise;

figure(1), imshow(f);
figure(2), hist(noise);
figure(3), imshow(uint8(noisy_image));
```

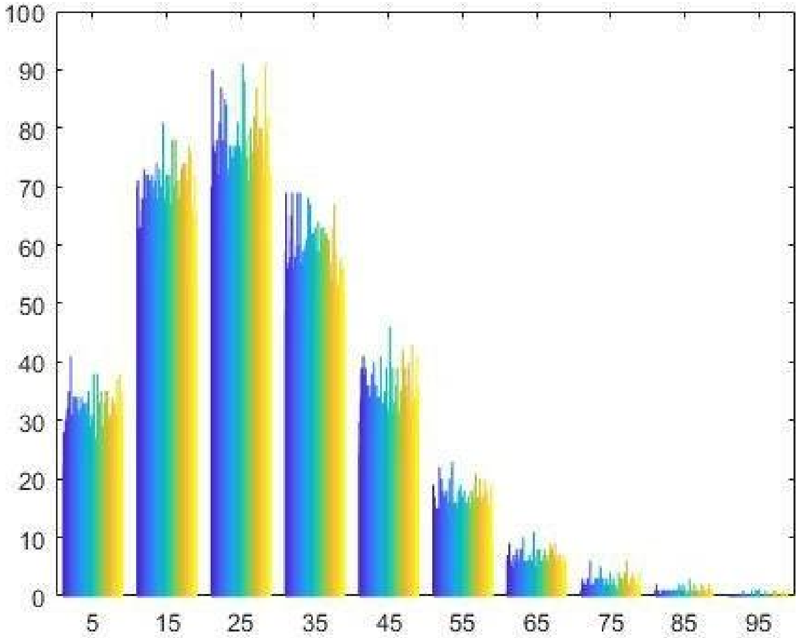
%Output:

% 1. Which noise do you want to add? Uniform = 1; Rayleigh = 2; Expo = 3: 1

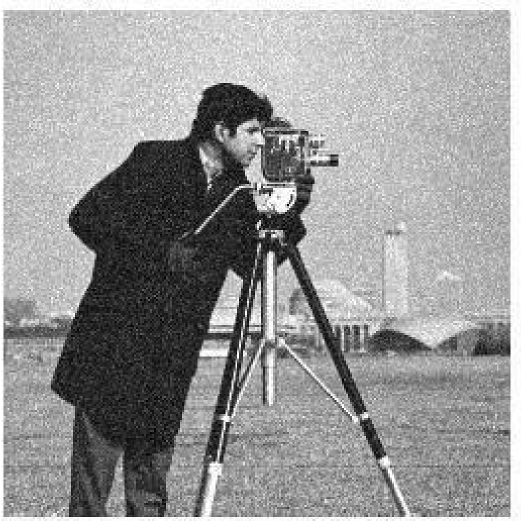




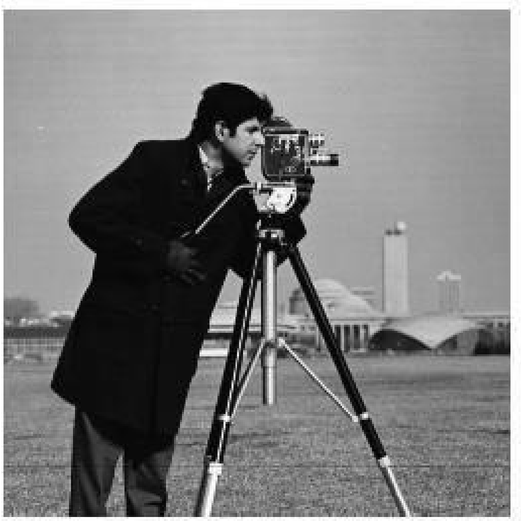
% 2. Which noise do you want to add? Uniform = 1; Rayleigh = 2; Expo = 3: 2

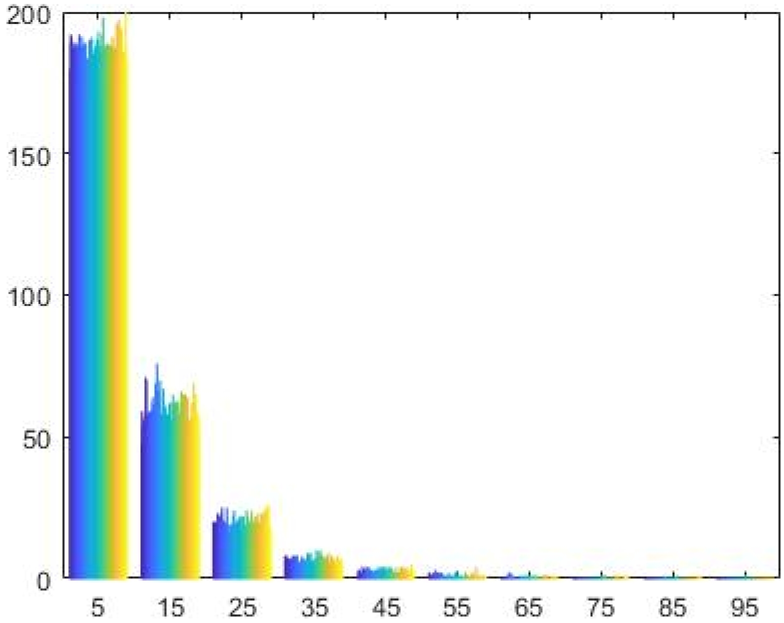


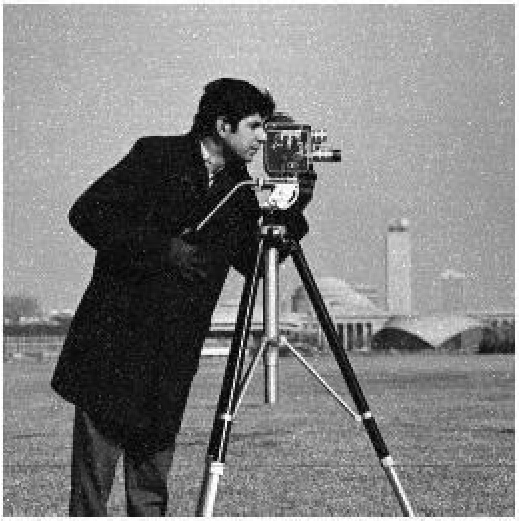
100
90
80
70
60
50
40
30
20
10
0
5
15
25
35
45
55
65
75
85
95



% 3 Which noise do you want to add? Uniform = 1; Rayleigh = 2; Expo = 3: 3

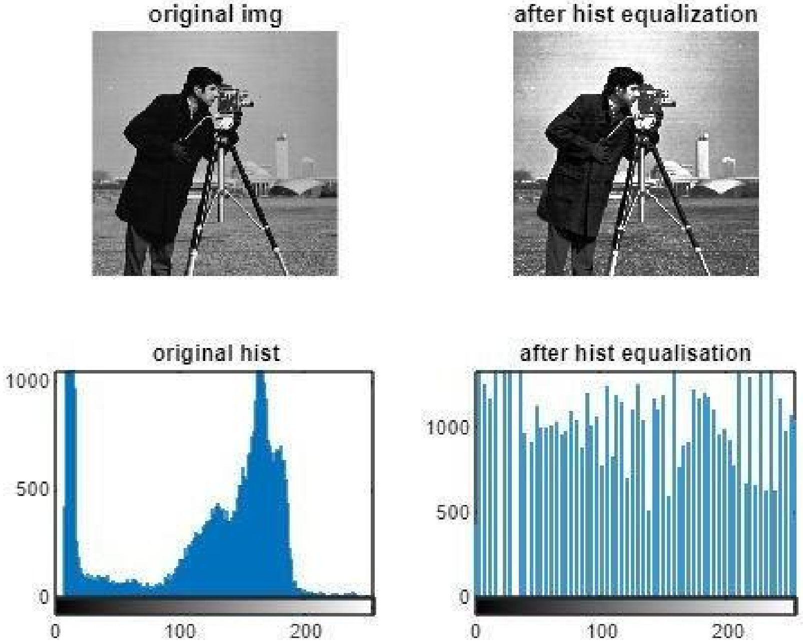



200
150
100
50
0
5
15
25
35
45
55
65
75
85
95



```
% Expt. No. 8
% Aim :-Histogram Equalisation (using readymade function)


 clc;
 close all;
 clear all;


 a = imread('cameraman.tif');
 % Histogram equalization using readymade function

 b = histeq(a);
 subplot(2,2,1), imshow(a), title('original img');
 subplot(2,2,2), imshow(b), title('after hist equalization');
 subplot(2,2,3), imhist(a), title('original hist');
 subplot(2,2,4), imhist(b), title('after hist equalisation');
```

Output:


original img
after hist equalization
original hist
1000
500
0
0
100
200
after hist equalisation
1000
500
0
0
100
200

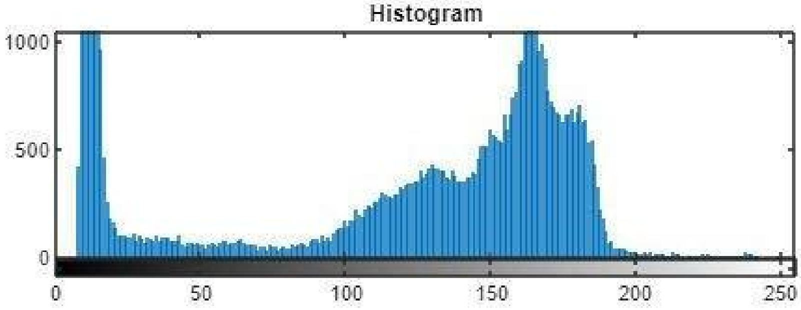
% Expt. No. 8

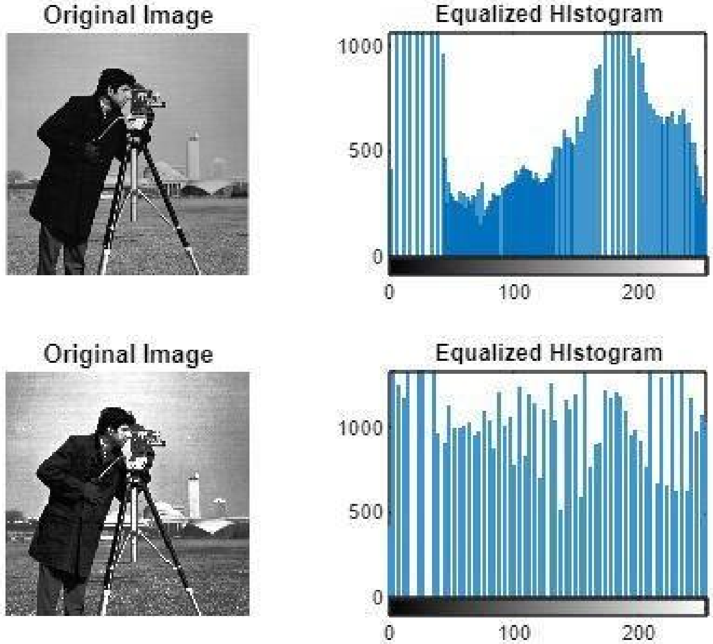
% Aim/ Title: Read an image, plot its histogram then do histogram equalization. Comment about the result. (Using User defined)

```
clc;

close  all;

clear all;


a = imread('cameraman.tif');

figure;

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

subplot(2,1,2); imhist(a); title('Histogram');


% Using readymade function

[mr,mc] = size(a);


% Declaration of all variables

him = uint8(zeros(mr,mc));

freq = zeros(256,1);

prob = zeros(256,1);

pdf = zeros(256,1);

cu = zeros(256,1);

op = zeros(256,1);

nopixels = mr*mc;

for i = 1:mr

   for j =1:mc
      v = a(i,j);

      freq(v+1) = freq(v+1)+1; prob(v+1)

      = freq(v+1)/nopixels;

   end end

sum=0;

n = 255;

for i = 1:size(prob);

      sum = sum+prob(i);
```

```
    cu(i) = sum;
        op(i) = round(cu(i)*n);
end
for  i = 1:mr
    for  j = 1:mc
        him(i,j) = op(a(i,j)+1);
        end
end
figure
subplot(2,2,1); imshow(a); title('Original Image');
subplot(2,2,2); imhist(him); title('Equalized HIstogram');

him1 = histeq(a);

subplot(2,2,3); imshow(him1); title('Original Image');

subplot(2,2,4); imhist(him1); title('Equalized HIstogram');
```

Output:




Original Image
Equalized HIstogram
1000
500
0
0
100
200
Original Image
Equalized HIstogram
1000
500
0
0
100
200

```
 % Expt. No. 9
 % Aim: Implement Huffman coding algorithm for image compression

 clc;
 close all;
 clear all;

 p = input('Enter the probabilities');
 Ps = sum(p);
m = fix(Ps);
display(m);
if(m == 1)
     S = sort(p,'descend');
     N = length(p);
     display(N);
     symbols = input('Enter the sysmbols of length N');
     display( symbols);
    [dict, avglen] = huffmandict( symbols, S);
    display(dict);
        temp = dict;
        for i = 1:length(temp)
        temp{i,2} = num2str(temp{i,2});

        end
   display( temp)
   display(avglen);
    sig = input('Enter the array of random source');
    encode = huffmanenco(sig,dict);
    display(encode);
    decode = huffmandeco(encode,dict);
    display(decode);
   else
   end
 N = length(p);
 Hx = 0;
for  i =1:N;
   Hx = Hx-(p(i)*(log2(p(i))));
end
disp('entropy')
display(Hx);
Efficency = Hx/avglen;
display(Efficency );
Percentage_Efficiency = 100*Efficency;
display(Percentage_Efficiency  );
```

Output:

```
Enter the probabilities [0.4 0.3 0.3]

m = 1



N = 3

Enter the sysmbols of length N {'a' 'f' 'g'}

symbols = 'a'  'f'        'g'

dict =  'a'     [         1]
        'f'     [1x2 double]
        'g'     [1x2 double]

temp = 'a'      '1'
        'f'           '0      1'
        'g'           '0      0'

avglen = 1.6000
```

```
Enter the array of random source{'a' 'f' 'g' 'f'}

encode =

     1     0     1     0     0     0     1


decode =

    'a'    'f'    'g'    'f'

entropy

Hx = 1.5710

Efficency = 0.9818

Percentage_Efficiency = 98.1844
```