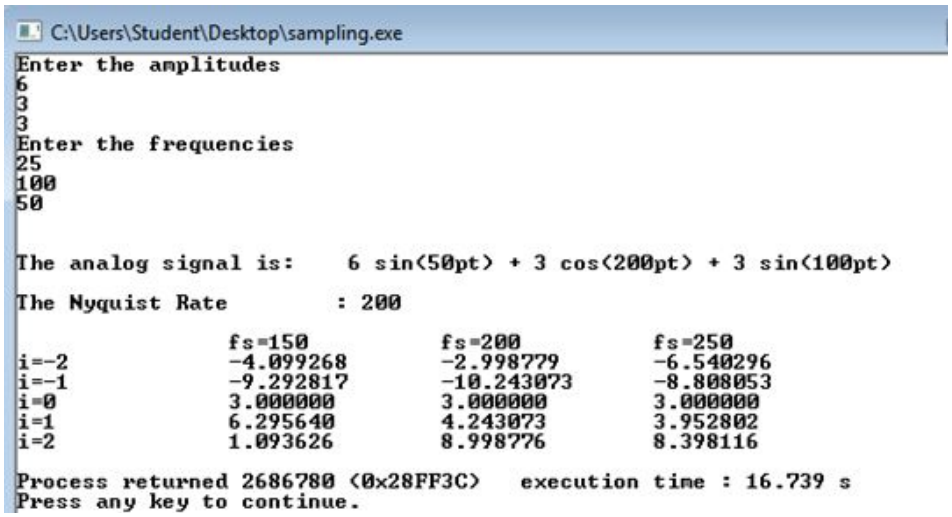
Experiment 1:

Perform a C/C++ program on Sampling of an analog CT signal (audio, speech , ECG etc.) and find its maximum frequency,Nyquist frequency and digital frequency. Analyse the number of digital samples/cycle and comment on your results for oversampling and undersampling. Plot your results on graph paper.

Experiment 1:

```
#include<math.h>
void main(){
        int i,a[30];
        int f[30],fs,k1,k2;
        float b1,b2,b3,p,q,r;
        printf("Enter the amplitudes\n");
        for(i=0;i<3;i++)
        scanf("%d",&a[i]);
        printf("Enter the frequencies\n");
        for(i=0;i<3;i++)
        scanf("%d",&f[i]);
printf("\n\nThe analog signal is:\t %d sin(%dpt) + %d cos(%dpt) + %d
sin(%dpt)",a[0],2*f[0],a[1],2*f[1],a[2],2*f[2]);
   fs=2*(f[0]>f[1]?(f[0]>f[2]?f[0]:f[2]):(f[1]>f[2]?f[1]:f[2]));
printf("\n\nThe Nyquist Rate\t: %d\n\n",fs);
        k1=fs-50;
        k2=fs+50;
        b1=(float)(2*3.142/k1);
        b2=(float)(2*3.142/fs);
        b3=(float)(2*3.142/k2);
   printf("\t\tfs=%d\t\tfs=%d\t\tfs=%d\n",fs-50,fs,fs+50);
        for(i=-2;i<3;i++){
        p=a[0]*sin(i*b1*f[0])+a[1]*cos(f[1]*i*b1)+a[2]*sin(i*b1*f[2]);
        q=a[0]*sin(i*b2*f[0])+a[1]*cos(f[1]*i*b2)+a[2]*sin(i*b2*f[2]);
 r=a[0]*sin(i*b3*f[0])+a[1]*cos(f[1]*i*b3)+a[2]*sin(i*b3*f[2]);
        printf("i=%d\t\t%3f\t%3f\t%3f\n",i,p,q,r);
        }
}
```

OUTPUT:



Experiment 2:

Perform a C/C++ program on Discrete AutoCorrelation of a 1-D DT signal and analyse its value at lag=0. write a note on its applications.

Experiment 2:

```
#include<stdio.h>
void convolution(int h[],int x[],int m,int n,int s);
void autocorrelation(int x[],int m,int s);
void crosscorrelation(int y[],int x[],int m,int n,int s1,int s2);
void faux_main();
main()
{
int x[15],h[15];
int i,m,n,s1,s2,s;
printf("choose: 1.convolution 2. correlation 3.exit");
do{
scanf("%d",&s);
switch(s)
{
case 1:printf("enter no. of input response x(n) and starting point :");
scanf("%d %d",&m,&s1);
printf("enter no. of impulse response h(n) and starting point :");
scanf("%d %d",&n,&s2);
printf("enter the value of x(n):");
for(i=0;i<m;i++)
{
scanf("%d",&x[i]);
}
printf("enter the value of h(n):");
for(i=0;i<n;i++)
{
scanf("%d",&h[i]);
}
convolution(h,x,m,n,s1+s2);break;
case 2:faux_main();break;
}
}while(s!=3);
}
#include<stdio.h>
int convolution(int h[],int x[],int m,int n,int s);
void autocorrelation(int x[],int m,int s)
{
int i;
int h[15];
for(i=0;i<m;i++)
h[i]=x[m-i-1];
printf("\nAuto correlation :\n");
```

```
int e=convolution(x,h,m,m,-(m-1));
printf("\nEnergy = %d",e);
}
void crosscorrelation(int y[],int x[],int m,int n,int s1,int s2)
{
int y2[15],k=0,i;
for(i=n-1;i>=0;i--)
y2[k++]=y[i];
printf("\nCrossCorrelation:\n");
int e=convolution(y2,x,m,n,s1-(s2+n-1));
printf("\n Power = %d",e);
}
void faux_main()
{
int s1,s2,m,n,x[15],y[15],s,i;
printf("choose : 1. cross correlation 2.auto correlation 3. exit");
do{
scanf("%d",&s);
switch(s)
{
case 1:printf("enter no. elements in x(n) and starting point :");
scanf("%d %d",&m,&s1);
printf("enter no. of elements in y(n) and starting point :");
scanf("%d %d",&n,&s2);
printf("enter the value of x(n):");
for(i=0;i<m;i++)
scanf("%d",&x[i]);
printf("enter the value of y(n):");
for(i=0;i<n;i++)
scanf("%d",&y[i]);
crosscorrelation(y,x,m,n,s1,s2);
break;
case 2: printf("enter no. elements in x(n) and starting point :");
scanf("%d %d",&m,&s1);
printf("enter the value of x(n):");
for(i=0;i<m;i++)
scanf("%d",&x[i]);
autocorrelation(x,m,s1);
break;
}
}while(s!=3);
}
```

OUTPUT:

```
choose: 1.convolution 2. correlation 3.exit
2
choose : 1. cross correlation 2.auto correlation 3. exit
1
enter no. elements in x(n) and starting point :4 -1
enter no. of elements in y(n) and starting point :4 -1
enter the value of x(n):1 2 3 4
enter the value of y(n):1 2 3 4

CrossCorrelation:
y[-3]=4 y[-2]=11        y[-1]=20        y[0]=30 y[1]=20 y[2]=11 y[3]=4
 Power = 30
2
enter no. elements in x(n) and starting point :4 -1
enter the value of x(n):1 2 3 4

Auto correlation :
y[-3]=4 y[-2]=11        y[-1]=20        y[0]=30 y[1]=20 y[2]=11 y[3]=4
Energy = 30
3
3

Process returned 3 (0x3)   execution time : 36.421 s
Press any key to continue.
```

Experiment 3:

Perform a C/C++ program on Discrete Convolution of a 1-D DT radar signal and its delayed response. Also find its cross-relation. Comment on your results.

Experiment 3:

MAIN:

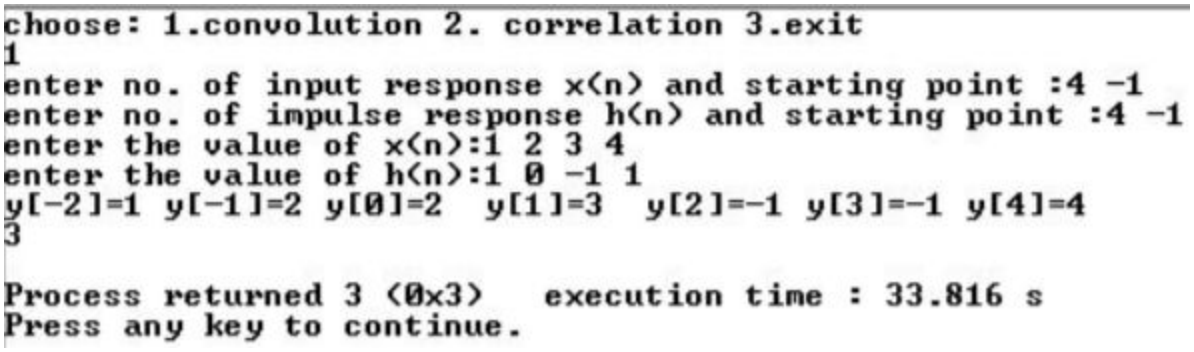
```
#include<stdio.h>
void convolution(int h[],int x[],int m,int n,int s);
void autocorrelation(int x[],int m,int s);
void crosscorrelation(int y[],int x[],int m,int n,int s1,int s2);
void faux_main();
main()
{
int x[15],h[15];
int i,m,n,s1,s2,s;
printf("choose: 1.convolution 2. correlation 3.exit");
do{
scanf("%d",&s);
switch(s)
{
case 1:printf("enter no. of input response x(n) and starting point :");
scanf("%d %d",&m,&s1);
printf("enter no. of impulse response h(n) and starting point :");
scanf("%d %d",&n,&s2);
printf("enter the value of x(n):");
for(i=0;i<m;i++)
{
scanf("%d",&x[i]);
}
printf("enter the value of h(n):");
for(i=0;i<n;i++)
{
scanf("%d",&h[i]);
}
convolution(h,x,m,n,s1+s2);break;
case 2:faux_main();break;
}
}while(s!=3);
}
```

CONVOLUTION:

```
#include<stdio.h>
int convolution(int h[],int x[],int m,int n,int s)
{
int i,j,r;
int y[15];
for(i=m;i<=m+n-1;i++)
{
```

```
x[i]=0;
}
for(i=n;i<=m+n-1;i++)
{
h[i]=0;
}
for(i=0;i<=m+n-1;i++)
{
y[i]=0;
for(j=0;j<=i;j++)
{
y[i]=y[i]+(x[j]*h[i-j]);
}
}
for(i=0;i<m+n-1;i++)
{
printf("y[%d]=%d\t",s++,y[i]);
if (s-1==0)
r=y[i];
}
return r;
}
```

OUTPUT:



Experiment 4:

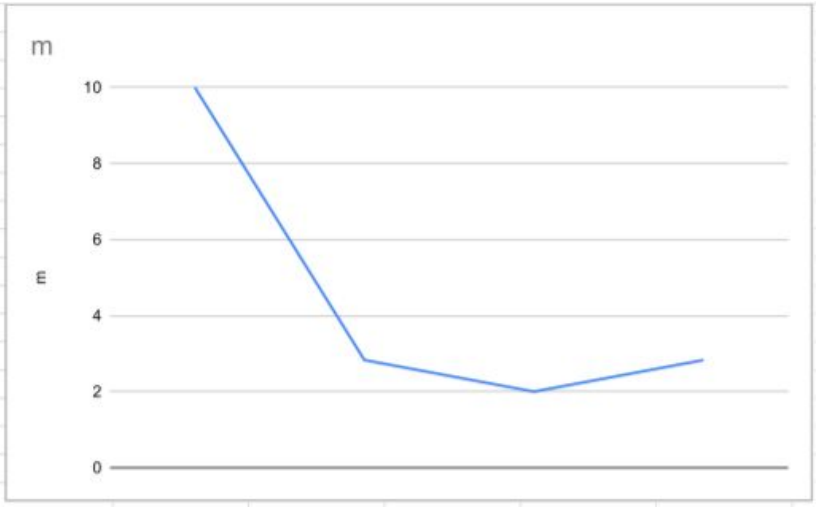
Perform a C/C++ program on 1-D 4-pt. Discrete Fourier Transform using the formula and verify your results using twiddle factor and matrix method. Find its significance.

Experiment 4:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int k,n,N;
float static X[100],real[100],imag[100],mag[100],phase[100];
printf("\nEnter N=");
scanf("%d",&N);
printf("Enter sequence x(n)\n");
for(n=0;n<N;n++)
{
scanf("%f",&X[n]);
}
for(k=0;k<N;k++)
{
real[k] = imag[k]=0.0;
for(n=0;n<N;n++)
{
real[k]=real[k]+X[n]*cos((2*M_PI*k*(n-N))/N);
imag[k]=imag[k]+X[n]*sin((2*M_PI*k*(n-N))/N);
}
imag[k]=imag[k]*(-1.0);
}
printf("\nThe %d point DFT X(k) of given sequence is:\n",N);
for(k=0;k<N;k++)
printf("\n%.2f + j %.2f",real[k],imag[k]);
for(k=0;k<N;k++){
mag[k]=sqrt(pow(real[k],2)+pow(imag[k],2));
phase[k]=atan(imag[k]/real[k]);
}
printf("\n");
printf("\n");
printf("\nMagnitude");
for(k=0;k<N;k++)
printf("\n%.2f",mag[k]);
printf("\n");
printf("\nPhase:");
for(k=0;k<N;k++){
printf("\n%.2f",(phase[k]*180)/M_PI);
}
}
```

```
C:\Users\Exam\Desktop\4dft.exe

Enter N=4
Enter sequence x(n)
1 2 3 4

The 4 point DFT X(k) of given sequence is:

10.00 + j -0.00
-2.00 + j 2.00
-2.00 + j -0.00
-2.00 + j -2.00

Magnitude
10.00
2.83
2.00
2.83

Phase:
-0.00
-45.00
0.00
45.00
Process returned 4 (0x4)   execution time : 16.889 s
Press any key to continue.
```

Magnitude	k
10	4
2.83	3
2	2
2.83	1



Experiment 5:

Perform a C/C++ program on 1-D 4-pt. Fast Fourier Transform of the above sequence and find the reduction in number of addition and multiplication.

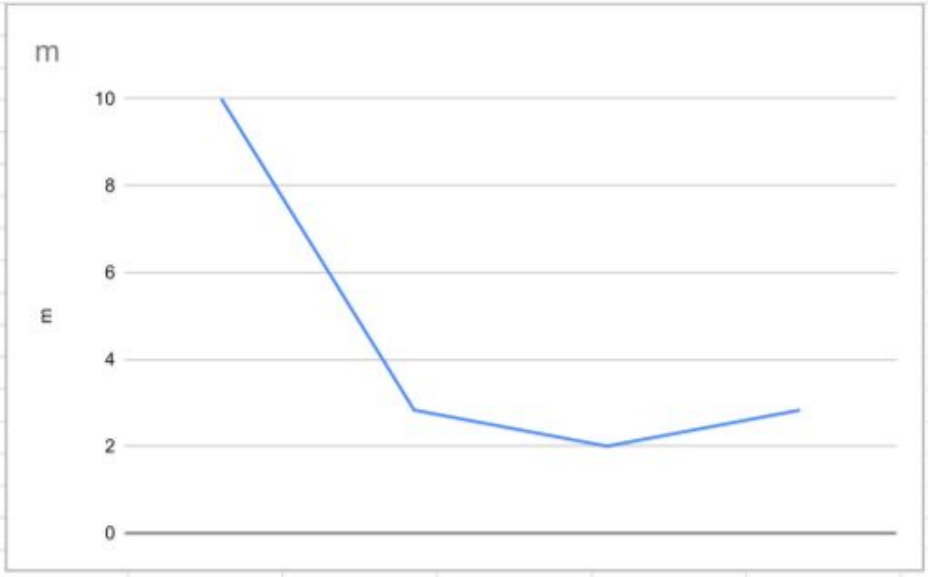
Experiment 5:

```
#include<stdio.h>
#include<stdlib.h>
int main(){
        int N, i=0;
        printf("Enter N : \n");
        scanf("%d",&N);
        printf("Enter x(n) values\n");
        int a[N];
        for(i=0; i<N;i++){
        scanf("%d", &a[i]);
        }
        int intervals[N];
        intervals[0] = (a[0]+a[2]);
        intervals[1] = a[1]+a[3];
        intervals[2] = a[0]-a[2];
        intervals[3] = a[1] - a[3];
        int finalvalsReal[N];
        int finalValsImgainary[N];
        finalvalsReal[0] = intervals[0] + intervals[1];
        finalvalsReal[1] = intervals[2] ;
        finalValsImgainary[1] = -intervals[3];
        finalvalsReal[2] = intervals[0] - intervals[1];
        finalvalsReal[3] = intervals[2];
        finalValsImgainary[3] = intervals[3];
        finalValsImgainary[0]=0;
        finalValsImgainary[2] = 0;
        for(i=0;i<N;i++){
        printf("FFT of arr[%d] is  %d + %dj\n", i,finalvalsReal[i],finalValsImgainary[i]);
        }
}
```

OUTPUT:

```
C:\Users\Exam\Desktop\fftfinal.exe
Enter N :
4
Enter x(n) values
1
2
3
4
FFT of arr[0] is  10 + 0j
FFT of arr[1] is  -2 + 2j
FFT of arr[2] is  -2 + 0j
FFT of arr[3] is  -2 + -2j

Process returned 0 (0x0)   execution time : 4.896 s
Press any key to continue.
```

Magnitude	k
10	4
2.83	3
2	2
2.83	1



Experiment 6:

Implementation of Negative, Gray level Slicing and Thresholding (Virtual lab) of a 2-D grey level image (IIT, Hyderabad)

Experiment 6:

```
clear all;
clc;
a=imread('xray.tif')
a=double(a);
[row col]=size(a);
LT=input('Enter the lower threshold value:');
UT=input('Enter the upper threshold value:');
for x=1:1:row
   for y=1:1:col
      if a(x,y)<=LT
         b(x,y)=0.5*a(x,y);
      else if a(x,y)<=UT
            b(x,y)=2*(a(x,y)-LT)+0.5*LT;
      else b(x,y)=0.5*(a(x,y)-UT)+0.5*LT+2*(UT-LT);
         end
      end
   end
end
subplot(2,1,1)
imshow(uint8(a))
title('Original Image');
subplot(2,1,2)
imshow(uint8(b))
title('Image after Contrast Stretching')
```

Output:

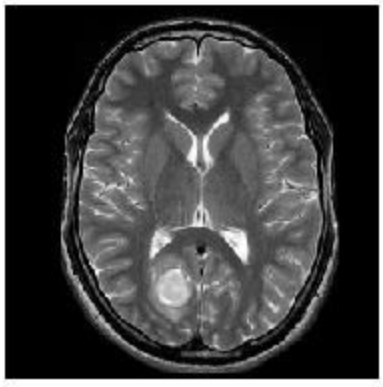
Experiment 7:

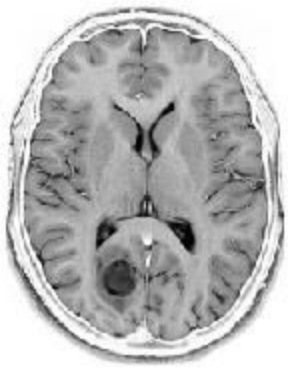
Implementation of Image negative, Gray level Slicing and Thresholding of a 2-D grey level image using MATLAB and evaluate its significances.

Experiment 7:
Negative:

```
clc
clear  all
close all
r=imread('C:\Users\Exam\Desktop\xray.png');
a=double(r);
L=256;
   [row col]=size(a);
   for i=1:1:row
      for j=1:1:col
         s(i,j)=(L-1)-a(i,j);
      end
   end
    figure(1);
   subplot(1,2,1);
   imshow(uint8(a));
   subplot(1,2,2);
   imshow(uint8(s));
```

Output:





Threshold:

```
clc
clear  all
close all
r=imread('C:\Users\Exam\Desktop\xray.png');
%a=rgb2gray(r);
for n=2:1:4
   a=r;
T= input('Enter:');
   [row col]=size(r);

   for i=1:1:row
      for j=1:1:col
         if a(i,j)<T
         a(i,j)=0;
         else
            a(i,j)=255;
         end
      end
   end
   figure(1);
    subplot(1,4,1);
   imshow(r);
    title('Original')
   subplot(1,4,n);
   imshow(a);
    title(T)
end
%   figure(1);
%     imshow(r);
%     imshow(a);
```

Grey level:

1. **With background:**

```
clc
clear all
p=imread('autumn.tif');
z=double(p);
[row col]=size(p);
for i= 1:1:row
   for j=1:1:col
      if(z(i,j)>100)&&(z(i,j)<140)
         z(i,j)=255;
      else
         z(i,j)=p(i,j);
      end
   end
end
figure(1);
imshow(p)
figure(2);
imshow(uint8(z))
title('With background');
```

2. **without:**

```
clc
clear all
p=imread('autumn.tif');
z=double(p);
[row col]=size(p);
for i= 1:1:row
   for j=1:1:col
      if(z(i,j)>100)&&(z(i,j)<250)
         z(i,j)=255;
      else
         z(i,j)=0;
      end
   end
end
figure(1);
         imshow(p)
         figure(2);
         imshow(uint8(z))
         title('without background');
```

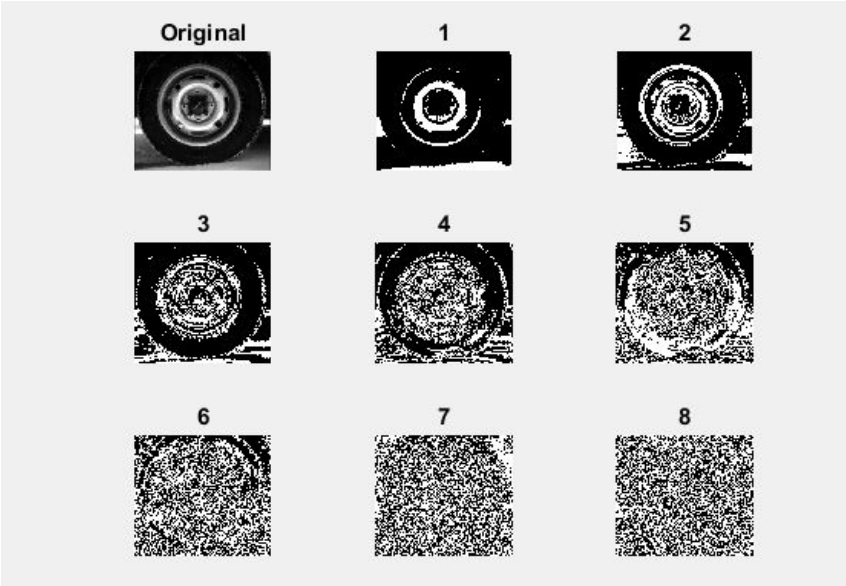
Experiment 8:

Implementation of Bit plane slicing for steganography of a 2-D grey level image using MATLAB

Experiment 8:
Bit Plane Extraction:

```
clear all
clc
a = imread('coins.png');
a = double(a);
[row, col] = size(a);
subplot(3,3,1);
imshow(uint8(a));
title('Original');
for i= 1:1:8
   for x=1:1:row
      for y=1:1:col
         c = dec2bin(a(x,y),8);
         d = c(i);
         w(x,y) = double(d);
         if w(x,y)==49
            w(x,y)=255;
         else
            w(x,y)=0;
         end
      end
   end
   subplot(3,3,(i+1));
   imshow(uint8(w));
   title(i);
end
```

Output:

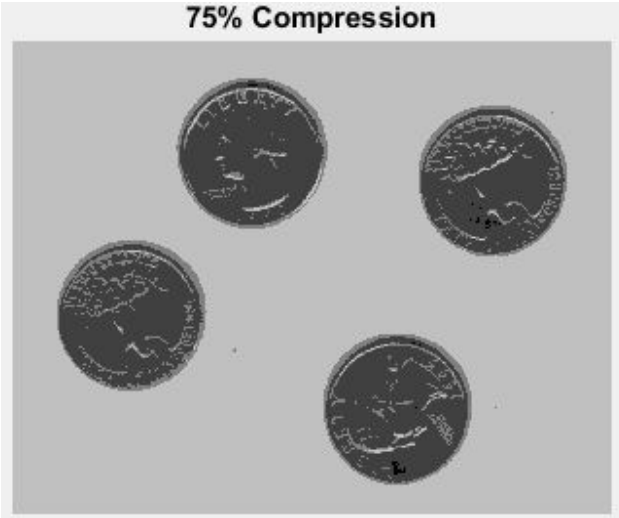


Bit compression:

1. **75% compression:**

```
clc
close all
A=imread('coins.png');
figure,imshow(A);
B=zeros(size(A));
B=bitset(B,7,bitget(A,7));
%bitset is used to set a bit at a specified position. bitget is used to get
%the bit at the specified position from all the pixels in the matrix A and
%therefore use bitset to set these bit values at the specified position in
%the matrix B
B=bitset(B,8,bitget(A,8));
B=uint8(B);
title('75% Compression');
figure
subplot(1,4,1);
imshow(B);
```

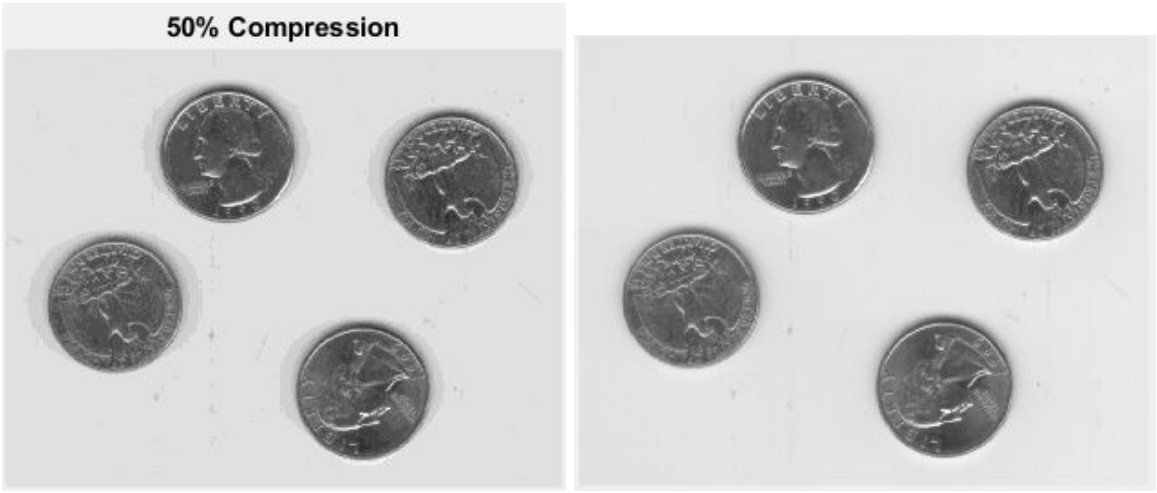
Output:



2. **50% Compression:**

```
clc
close all
A=imread('coins.png');
figure,imshow(A);
B=zeros(size(A));
B=bitset(B,8,bitget(A,8));
B=bitset(B,7,bitget(A,7));
B=bitset(B,6,bitget(A,6));
B=bitset(B,5,bitget(A,5));
B=uint8(B);
figure,imshow(B);
title('50% Compression');
```

Output:



Steganography and retrieval:

```
clear all
clc
close all
i1=imread('cameraman.tif');
i1=double(i1);
[row col]=size(i1);
a=imread('C:\Users\Exam\Pictures\help.bmp');
i2=imresize(a,[row col]);
i2=double(i2);
for x=1:1:row
   for y=1:1:col
      t1=dec2bin(i1(x,y),8);
      t2=dec2bin(i2(x,y),8);
      t1(8)=t2(1);
      t1(7)=t2(2);
      t1(6)=t2(3);
      t1(5)=t2(4);
      t1(4)=t2(5);
      stego(x,y)=bin2dec(t1);
   end
end
stego=uint8(stego);
imwrite(stego,'newimage.tif')
figure(1)
%%subplot(2,2,1)
imshow(uint8(i1))
title('CARRIER IMAGE')
%%subplot(2,2,2)
figure(2)
imshow(uint8(i2))
title('SECRET DATA')
%%subplot(2,2,3)
figure(3)
imshow(stego)
title('STEGO IMAGE')
i=stego;
i=double(i);
[row col]=size(i);
for x=1:1:row
   for y=1:1:col
      t=dec2bin(i(x,y),8);
      if(t(8)=='0')
```

```
            t='00000000';
            secret_data(x,y)=bin2dec(t);
        else
            t='11111111';
            secret_data(x,y)=bin2dec(t);
        end
    end
end
%%subplot(2,2,1)
figure(4)
imshow(uint8(secret_data));
title('RETRIEVED IMAGE')
```

CARRIER IMAGE

SECRET DATA

help

STEGO IMAGE

RETRIEVED IMAGE

help

Experiment 9:

Implementation of Histogram equalization of a 2-D grey level image using MATLAB and analyse the need of this process.

Experiment 9:

```
clear all
clc
a = imread('pout.tif');
a = double(a);
big = max((max(a)));
[row,col]=size(a);
C = row*col;
h = zeros(1,300);
z = zeros(1,300);
for n = 1:1:row
    for m = 1:1:col
        if a(n,m)==0
            a(n,m)=1;
        end
    end
end
 for n = 1:1:row
        for m = 1:1:col
            t = a(n,m);
            h(t)=  h(t)+1;
        end
end
    pdf = h/C;
   cdf(1)=pdf(1);
for x=2:1:big
       cdf(x)=pdf(x)+cdf(x-1);
   end
  new = round(cdf*big);
  new = new+1;
  for p = 1:1:row
      for q =1:1:col
          temp = a(p,q);
          b(p,q) = new(temp);
          t = b(p,q);
          z(t) =  z(t)+1;
      end
  end
  b =  b-1;
  figure(1),imshow(uint8(a))
  figure(2), bar(h)
  figure(3), imshow(uint8(b))
  figure(4), bar(z)
```

Output:

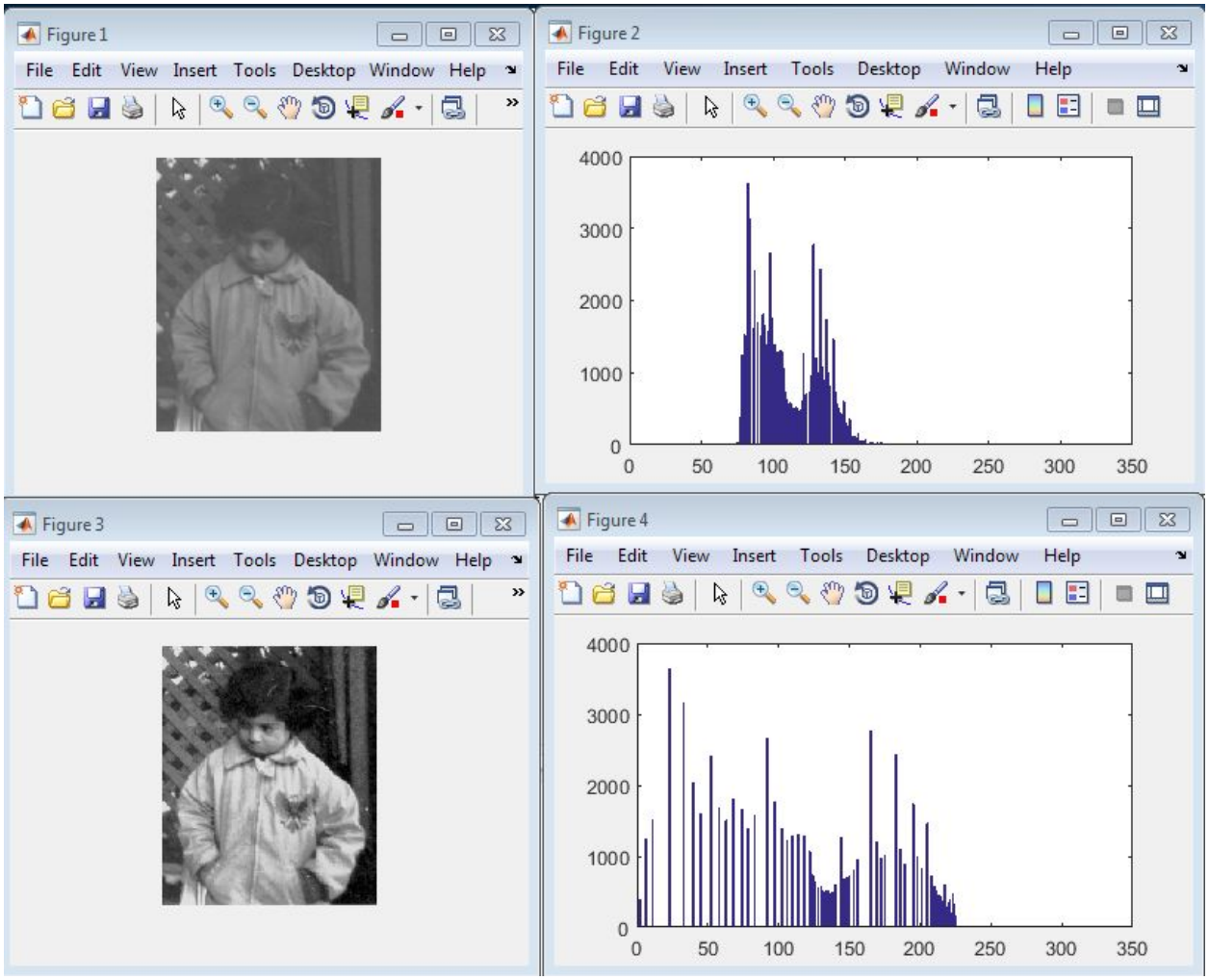


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
File Edit View Insert Tools Desktop Window Help
4000
3000
2000
1000
0
0 50 100 150 200 250 300 350
Figure 3
File Edit View Insert Tools Desktop Window Help
Figure 4
File Edit View Insert Tools Desktop Window Help
4000
3000
2000
1000
0
0 50 100 150 200 250 300 350

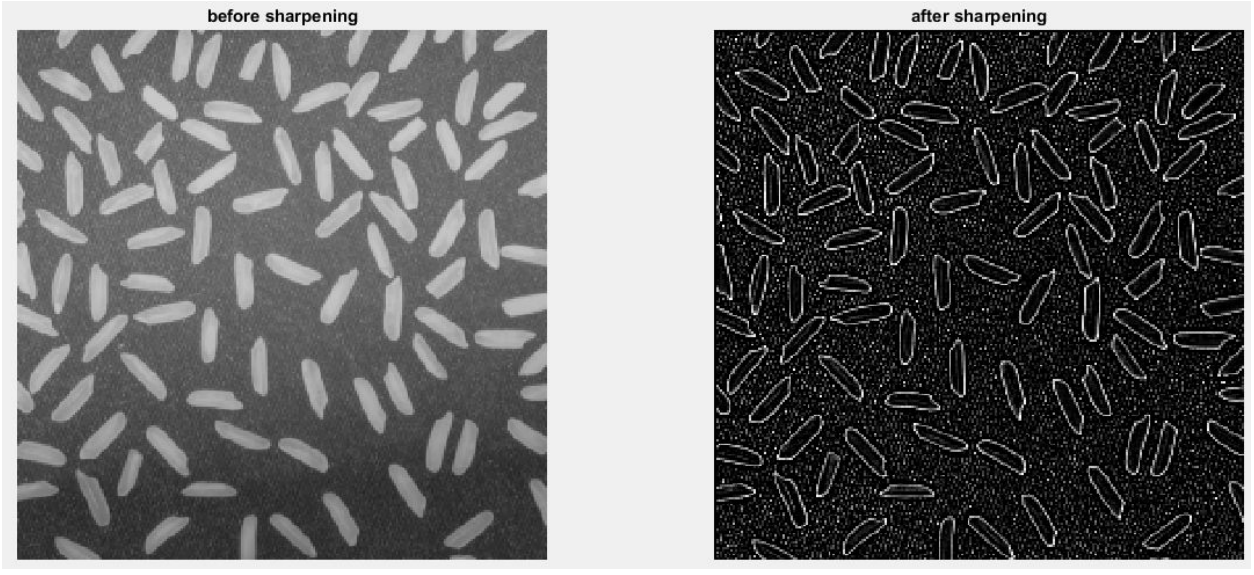
Experiment 10:

Implementation of Image smoothing and sharpening of a grey level image using MATLAB. Analyse the results.

Experiment 10:

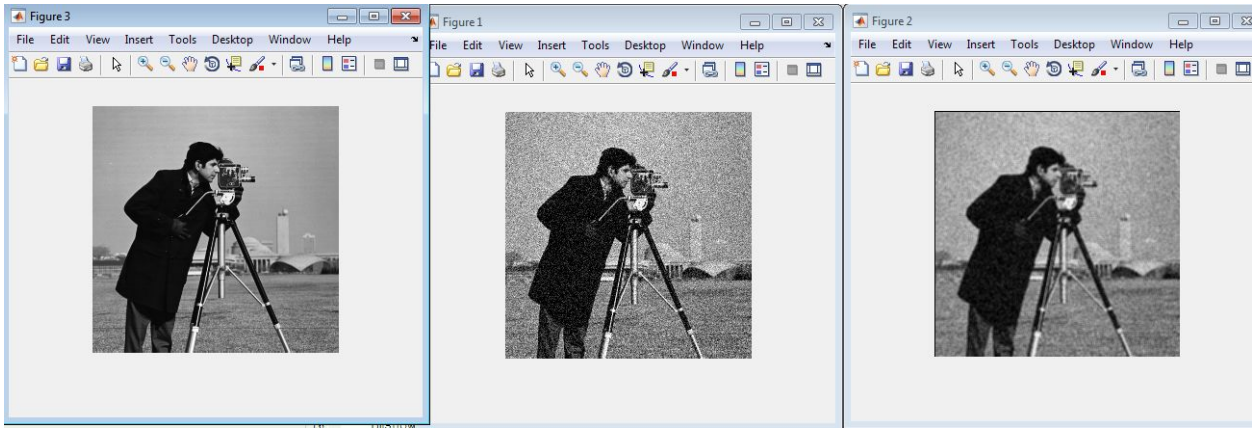
```
clear all
clc
aa = imread('rice.png');
a = double(aa);
[row, col] = size(a);
w = [-1 -1 -1;-1 8 -1;-1 -1 -1]
for x=2:1:row-1
   for y=2:1:col-1
      a1(x,y)=
w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)*a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)*a(x,y+1)+w(7)*a(
x+1,y-1)+w(8)*a(x+1,y)+w(9)*a(x+1,y+1);
   end
end
figure(1)
subplot(1,2,1)
imshow(uint8(a))
title('before sharpening')
subplot(1,2,2)
imshow(uint8(a1))
title('after sharpening')
```

OUTPUT:



Low pass filter

```
clear all
clc
aa=imread('cameraman.tif');
a=double(aa);
ab=imnoise(aa,'gaussian');
a=double(ab);
w=[1 1 1;1 1 1;1 1 1]/9
    [row col]=size(a);
   for x=2:1:row-1
       for y=2:1:col-1
          a1(x,y) =
w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)*a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)*a(x,y+1)+...
             w(7)*a(x+1,y-1)+w(8)*a(x+1,y)+w(9)*a(x+1,y+1);
       end
   end
figure(1)
imshow(uint8(a));
figure(2)
imshow(uint8(a1));
figure(3)
imshow(uint8(aa));
```



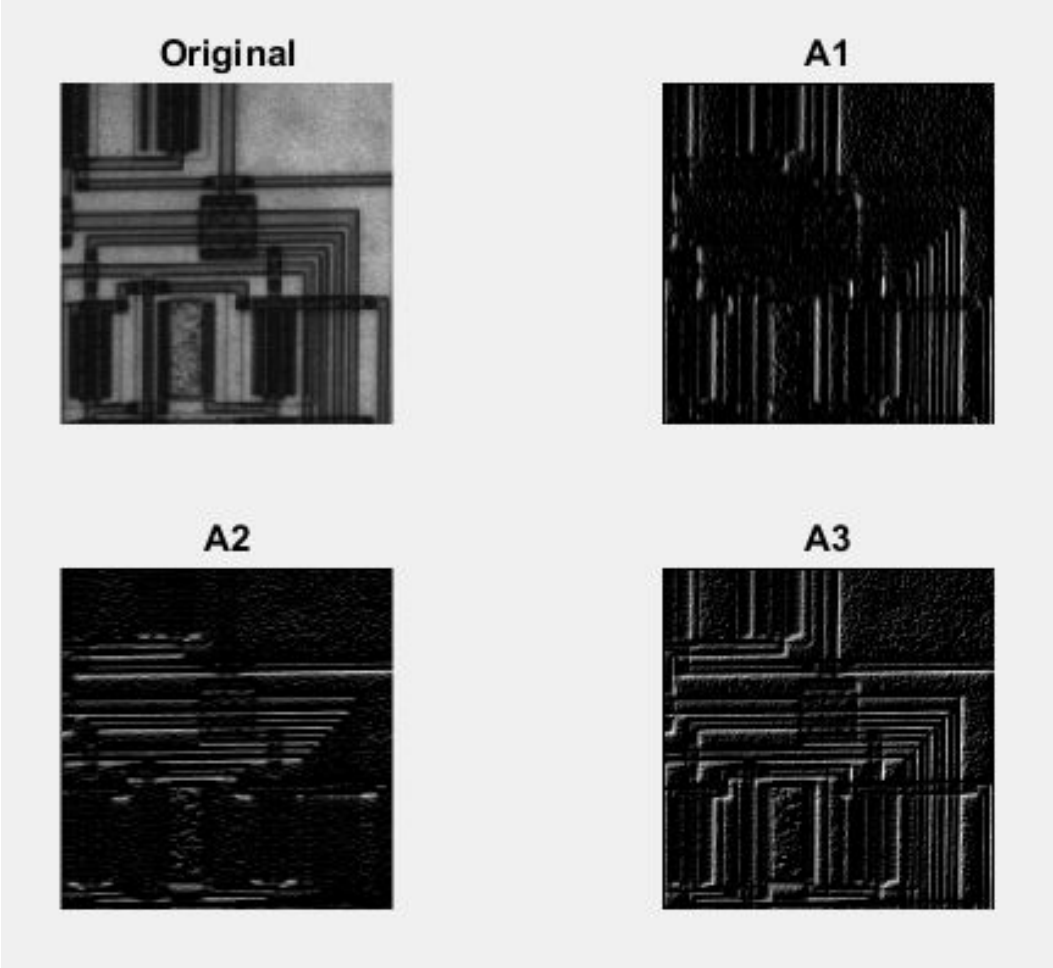
Experiment 11:

Implementation of Edge detection using sobel and previtt masks of a grey level image using MATLAB and analyse its applications in real life.

Experiment 11:

Prewitt:

```
clear all
clc
aa = imread('circuit.tif');
a = double(aa);
[row, col] = size(a);
w2 = [-1, 0, 1; -1 0 1; -1 0 1];
w1 = [-1 -1 -1; 0 0 0; 1 1 1];
for x=2:1:row-1
   for y = 2:1:col-1
      a1(x,y) = w1(1)*a(x-1,y-1)+w1(2)*a(x-1,y)+w1(3)*a(x-1,y+1)+w1(4)*a(x,y-1)+...
      w1(5)*a(x,y)+w1(6)*a(x,y+1)+w1(7)*a(x+1,y-1)+w1(8)*a(x+1,y)+w1(9)*a(x+1,y+1);

      a2(x,y) = w2(1)*a(x-1,y-1)+w2(2)*a(x-1,y)+w2(3)*a(x-1,y+1)+w2(4)*a(x,y-1)+...
      w2(5)*a(x,y)+w2(6)*a(x,y+1)+w2(7)*a(x+1,y-1)+w2(8)*a(x+1,y)+w2(9)*a(x+1,y+1);
   end
end
a3 = a1+a2;
figure(1)
subplot(2,2,1)
imshow(aa)
title('Original')
subplot(2,2,2)
imshow(uint8(a1))
title('A1')
subplot(2,2,3)
imshow(uint8(a2))
title('A2')
subplot(2,2,4)
imshow(uint8(a3))
title('A3')
```



Sobel:

```
clear all
clc
aa=imread('cameraman.tif');
a=double(aa);
[row, col]=size(a);
w1=[-1 -2 -1; 0 0 0; 1 2 1 ];
w2=[-1 0 1; -2 0 2; -1 0 1];
for x=2:1:row-1;
   for y=2:1:col-1;

a1(x,y)=w1(1)*a(x-1,y-1)+w1(2)*a(x-1,y)+w1(3)*a(x-1,y+1)+w1(4)*a(x,y-1)+w1(5)*a(x,y)+w1(6)*a
(x,y+1)+w1(7)*a(x+1,y-1)+w1(8)*a(x+1,y)+w1(9)*a(x+1,y+1);

a2(x,y)=w2(1)*a(x-1,y-1)+w2(2)*a(x-1,y)+w2(3)*a(x-1,y+1)+w2(4)*a(x,y-1)+w2(5)*a(x,y)+w2(6)*a
(x,y+1)+w2(7)*a(x+1,y-1)+w2(8)*a(x+1,y)+w2(9)*a(x+1,y+1);
   end
```

```
end
a3=a1+a2;
figure(1)
subplot(2,2,1)
imshow(aa)
title('Original')
subplot(2,2,2)
imshow(uint8(a1))
title('A1')
subplot(2,2,3)
imshow(uint8(a2))
title('A2')
subplot(2,2,4)
imshow(uint8(a3))
title('A3')
```