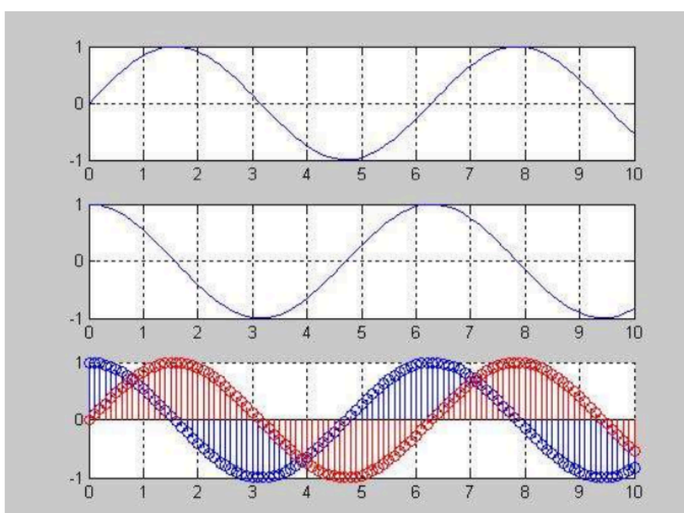
DSP LAB

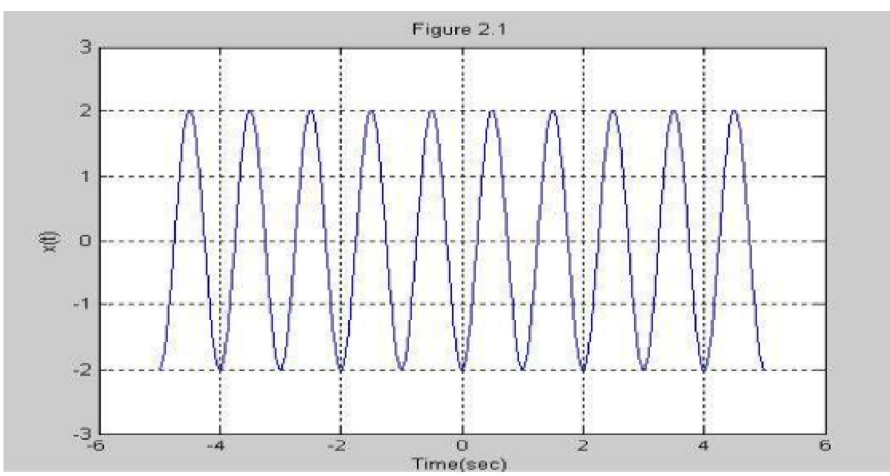
Experiment 1 (PLOTTING)

```
x = [0:0.1:10];
y = sin (x);
z = cos (x);
subplot (3,1,1);
plot (x,y);
grid on;
subplot (3,1,2);
plot (x,z);
grid on;
hold on;
subplot (3,1,3);
stem (x,z);
grid on;
hold on;
subplot (3,1,3);
stem (x,y, ,'r');
```



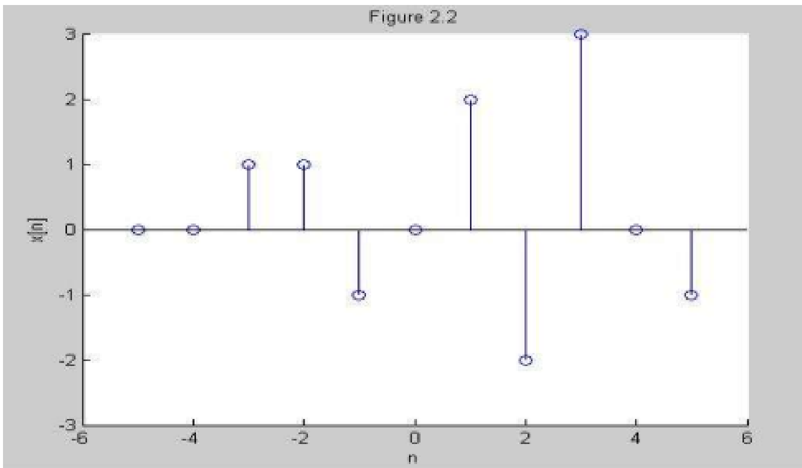
Experiment 2 (Generating a Signal)

```
% Generation of discrete time signals
% 2sin(2πτ-π/2)
T = [-5:0.01:5];
x=2*sin((2*pi*t) - (pi/2));
plot(t,x)
grid on;
axis ([-6 6 -3 3])
ylabel ('x(t)')
xlabel ('Time(sec)')
title ('Figure 2.1')
```



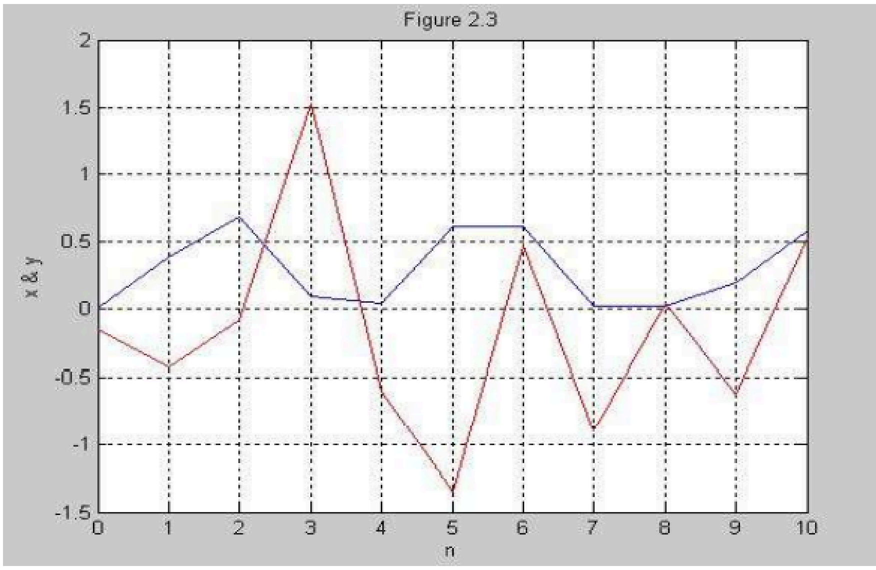
Experiment 3 (Generating a Signal)

```
% Generation of discrete time signals
n = [-5:5];
x = [0 0 1 1 -1 0 2 -2 3 0 -1];
stem (n,x);
axis ([-6 6 -3 3]);
xlabel ('n'); ylabel
('x[n]'); title
('Figure 2.2');
```



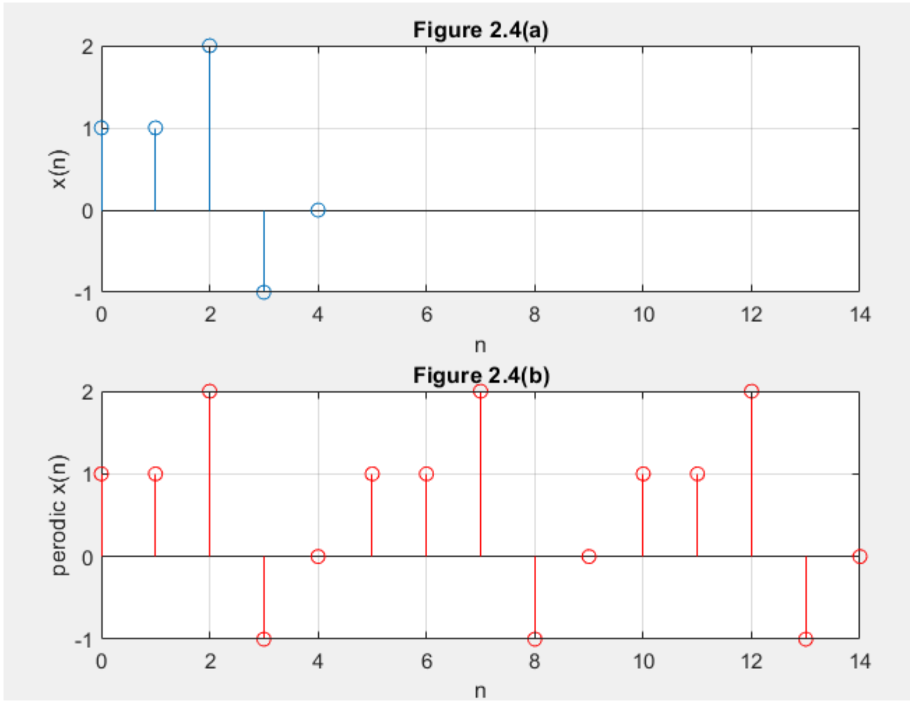
Experiment 4 (Generating a Signal)

```
%Generation of random sequence
n = [0:10];
x = rand (1, length (n));
y = randn (1, length (n));
plot (n,x) ;
grid on;
hold on;
plot(n,y,'r');
ylabel ('x & y')
xlabel ('n')
title ('Figure 2.3');
```



Experiment 5 (Generating a discrete periodic signal Signal)

```
n = [0:4];
x = [1 1 2 -1 0];
subplot (2,1,1);
stem (n,x);
grid on;
axis ([0 14 -1 2]);
xlabel ('n');
ylabel ('x(n)');
title ('Figure 2.4(a)');
xtilde = [x,x,x];
length_xtilde = length (xtilde);
n_new = [0:length_xtilde-1];
subplot (2,1,2);
stem (n_new, xtilde,'r');
grid on;
xlabel ('n');
ylabel ('perodic x(n)');
title ('Figure 2.4(b)');
```



Experiment 6 (Generating Square wave) using loop

```
clear;
clc;
n = input ('Insert the value of odd n:');
t = 0:.001:1;
sum=0;
for f=1:2:n
w=sin(2*pi*f*t);
sum=sum+w;
end
subplot(1,1,1)
plot(t,sum)
grid on;
```

Experiment 7 (Generating Unit Step Discrete Time Signal)

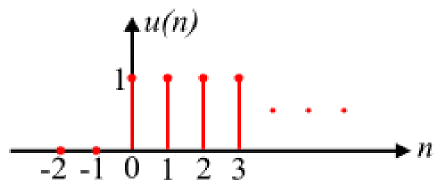
Experiment Name: Generating and Plotting Unit Step Discrete Time Signal.

Discrete Time Unit Step Signal:

It is denoted by u[n]. Mathematically, the discrete-time unit step signal or sequence u[n] is defined as follows –

$$u[n] = \{1 \quad for\ n \geq 0\ 0 \quad for\ n < 0$$

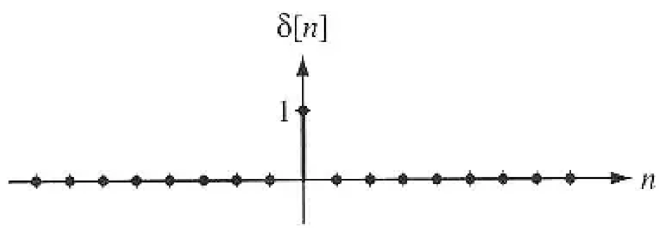
The graphical representation of the discrete-time unit step signal u[n] is shown in the following figure:




```
%Generating and Plotting Unit Step Discrete Time Signal.
clc; %clears the command window
clear all; %clears the current variables which are being used
close all; %close programs that are running behind in MATLAB

N=input('Enter the range: ');
n=-N:1:N;
y= [zeros(1,N),1,ones(1,N)];
stem(n,y);
axis([-(N+1) N+1 -0.5 1.5]); % [-x x -y y]
xlabel('Time');
ylabel('Amplitude of Y');
title('Generating Unit Step Function');
```

Experiment 8 : Unit impulse signal is mathematically defined as,

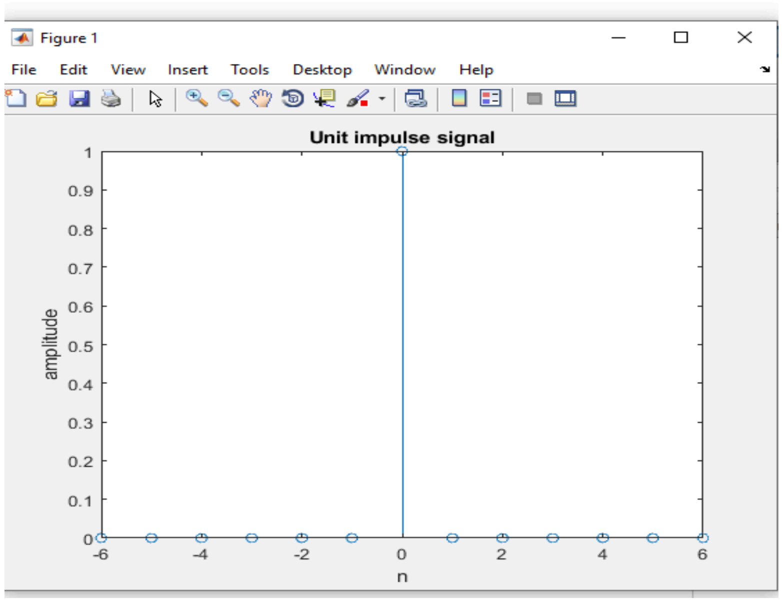
$$\boxed{\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}}$$



Code of implementation of impulse signal in Matlab:

```
%Code of implementation of impulse signal in Matlab:
clc;
clear all;
m1=input('enter the value of x-axis in negative side:');
m2=input('enter the value of x-axis in positive side:');
n=m1:m2;
x=(n==0);%it works as if statement like n=-5:5( 0 0 0 0 0 1 0 0 0 0 0 0)
stem(n,x);
xlabel('n');
ylabel('amplitude');
title('Unit impulse signal');
```

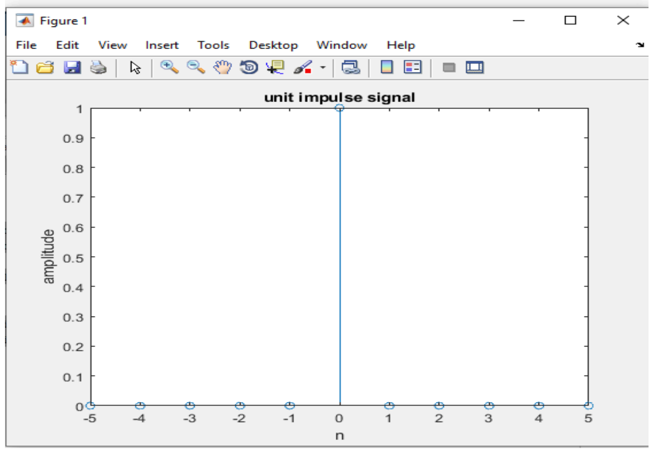
Output:



The unit impulse can be implemented in different way:

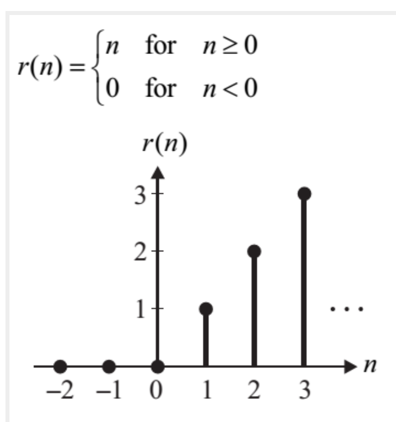
```
clc;
clear all;
close all;
m1=input('enter the value of x-axis in negative side:');
m2=input('enter the value of x-axis in positive side:');
n=-m1:m2;
d=[zeros(1,m1) 1 zeros(1,m2)];
stem(n,d);
xlabel('n');
ylabel('amplitude');
title('unit impulse signal');
```

Output:



Experiment 9 Generating and plotting ramp discrete time signal.

The discrete time unit ramp signal is that function which starts from n = 0 and increases linearly. It is denoted by r(n). It is signal whose amplitude varies linearly with time n. mathematically; the discrete time unit ramp sequence is defined as –



Code:

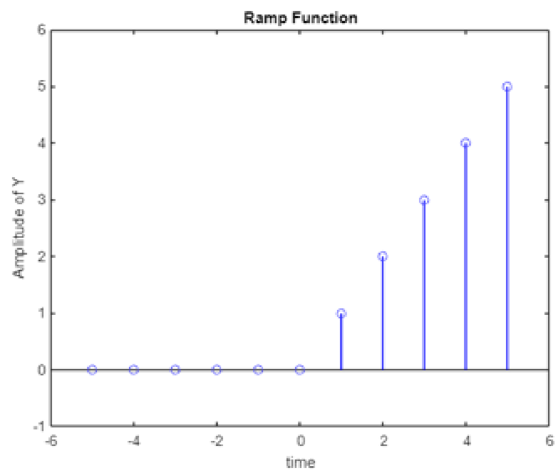
```
close all;
clear all;
clc;

n1= input ('Enter lower limit');
```

```
n2= input ('Enter upper limit');
n= n1: 1: n2;
x=n.*[n>=0]; % creates a ramp function
stem (n, x, 'b');
axis([(n1-1) (n2+1) -1 (n2+1)]);   % -x,x,-y,y
title (' Ramp Function ');
xlabel ('time');
ylabel ('Amplitude of Y');
```

Input:

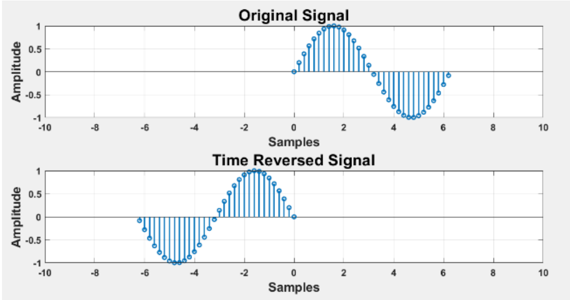
Enter lower limit
-5
Enter upper limit
5

Output:



Experiment 10 (Time reversal using a discrete sinusoidal function [use of fliplr( ) and values of x-axis(angle) in radian)

```
%Time reversal using a function (sinusoidal function
angle in radian)

close all
clc
t1=0:0.2:2*pi; %values of x-axis in radian
x1=sin(t1);    %values of y-axis
x2=fliplr(x1); %fliplr() -> this function gives the
flipped result;
              %lr means left right ...flipud() ud
means up down
t2= -fliplr(t1); % time values must be flipped and
negated
subplot(2,1,1)
stem(t1,x1,'LineWidth',2)
xlim([-10 10])
ylim([-1.5 1.5])
title('\bf\fontsize{15}Original Signal')
xlabel('\bf\fontsize{15}Samples')
ylabel('\bf\fontsize{10}Amplitude')
grid on;
subplot(212)
stem(t2,x2,"r",'LineWidth',2)
xlim([-10 10])
ylim([-1.5 1.5])
title('\bf\fontsize{15}Time Reversed Signal')
xlabel('\bf\fontsize{15}Samples')
ylabel('\bf\fontsize{10}Amplitude')
grid on;
```



Experiment 11 Time reversal using a discrete sinusoidal function [use of fliplr()and values of x-axis(angle) in degree]

```
%Time reversal using a function (sinusoidal function
angle in degree)

close all
clc
t1=0:10:360; %values of x-axis in degree
x1=sind(t1); % values of y axis
x2=fliplr(x1); %fliplr() -> this function gives the
flippefd result;
              %lr means left right ...flipud() ud
means up down
t2= -fliplr(t1); % time values must be flipped and
negated
subplot(211)
stem(t1,x1,'LineWidth',2)
xlim([-400 400])
ylim([-1.5 1.5])
title('\bf\fontsize{15}Original Signal')
xlabel('\bf\fontsize{15}Samples')
ylabel('\bf\fontsize{10}Amplitude')
grid on;
subplot(212)
stem(t2,x2,'r','LineWidth',2)
xlim([-400 400])
```

```
ylim([-1.5 1.5])
title('\bf\fontsize{15}Time Reversed Signal')
xlabel('\bf\fontsize{15}Samples')
ylabel('\bf\fontsize{10}Amplitude')
grid on;
```

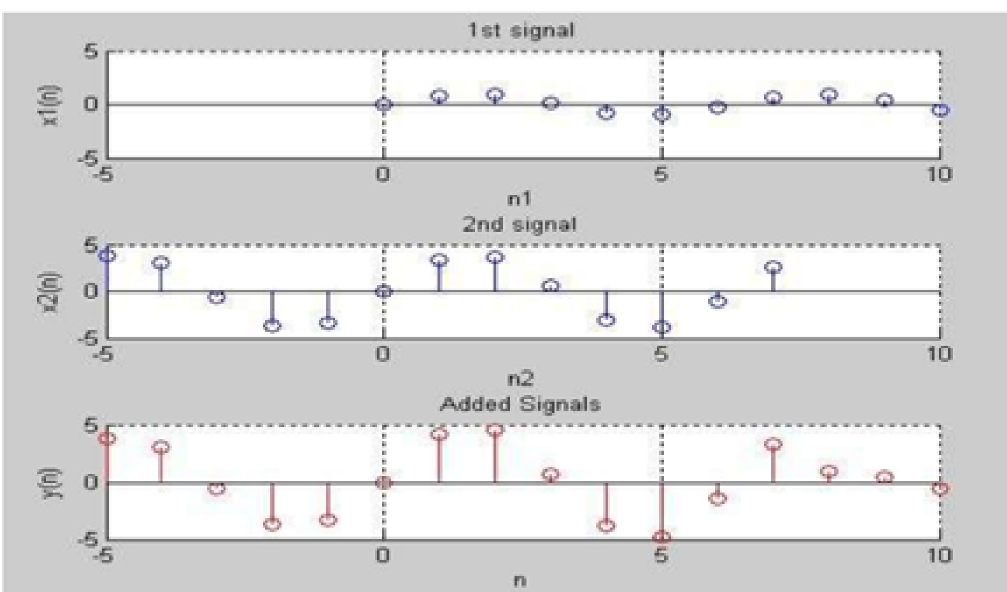
Experiment 12 (Signal Addition)

addition.m ->

```
clear all;
clc;
x1=[-5 -4 -3 -2 -1 0];
y1=[2 5 4 6 3 5];
x2=[-2 -1 0 1 2];
y2=[8 9 2 5 6];
% Draw the second signal.
subplot(3,1,1);
stem(x1,y1);
grid on;
grid minor;
axis([-10 10 -15 15]);
% Draw the second signal.
subplot(3,1,2);
stem(x2,y2);
grid on;
grid minor;
axis([-10 10 -15 15]);
n=min(min(x1),min(x2)):1:max(max(x1),max(x2)); % scaling
% This function is for the addition the two signal .
[y] = add_function(n,x1,x2,y1,y2);
% This is for the plot the added signal.
subplot(3,1,3);
stem(n,y);
grid on;
grid minor;
axis([-10 10 -15 15]);
```

add_function.m ->

```
function[y] = add_function(n,x1,x2,y1,y2)

m1=zeros(1,length(n));
m2=zeros(1,length(n));
temp=1;
for i=1:length(n)
   if(n(i)>=min(x1) & n(i)<=max(x1))
      m1(i)=y1(temp);
      temp=temp+1;
   else
      m1(i)=0;
   end
end
temp=1;
for i=1:length(n)
   if(n(i)>=min(x2) & n(i)<=max(x2))
      m2(i)=y2(temp);
      temp=temp+1;
   else
      m2(i)=0;
   end
end

y=m1+m2;
```



Experiment 13 (Signal Multiplication)

—-------------------

<u>Multiplicaton.m -></u>

—-------------------

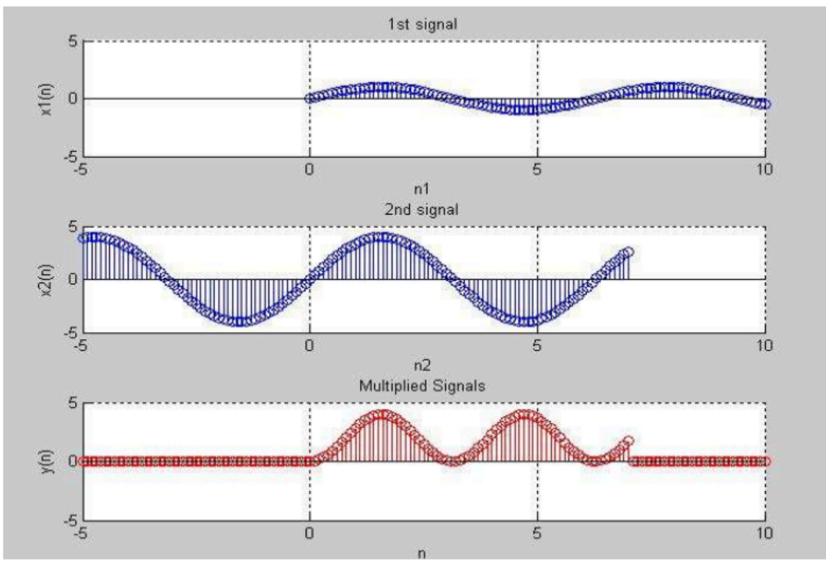
```
% Clearing existing figures, workspace, and command window
clc;
clear all;
close all;

% Define the first signal (x1, y1)
x1 = [0:0.1:10];
y1 = sin(x1);

% Define the second signal (x2, y2)
x2 = [-5:0.1:7];
y2 = 4*sin(x2);

% Plotting the graph of the first signal (x1, y1)
subplot(3,1,1);
stem(x1, y1);
grid on;
grid minor;
axis([-5 10 -5 5]);
title('Signal 1: (x1, y1)');

% Plotting the graph of the second signal (x2, y2)
subplot(3,1,2);
stem(x2, y2);
grid on;
grid minor;
axis([-5 10 -5 5]);
title('Signal 2: (x2, y2)');

% Finding the new range for the combined signal
n = min(min(x1), min(x2)):0.1:max(max(x1), max(x2));

% Calling the multiplication function
[m] = mul_function(n, x1, y1, x2, y2);
```

```
% Plotting the graph of the multiplied signal (n, m)
subplot(3,1,3);
stem(n, m, 'r');
grid on;
grid minor;
axis([-5 10 -5 5]);
title('Multiplication Result: (n, m)');
```

—-------------------
mul_function.m ->
—--------------------

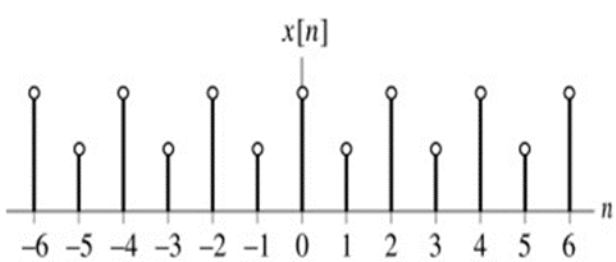
```
function [m] = mul_function(n, x1, y1, x2, y2)
% Function to multiply two signals

% Initialize arrays for the two signals
m1 = zeros(1, length(n));
m2 = m1;

% Loop to fill the first array (m1)
temp = 1;
for i = 1:length(n)
    if (n(i) >= min(x1) && n(i) <= max(x1))
        m1(i) = y1(temp);
        temp = temp + 1;
    else
        m1(i) = 0;
    end
end

% Loop to fill the second array (m2)
temp = 1;
for i = 1:length(n)
    if (n(i) >= min(x2) && n(i) <= max(x2))
        m2(i) = y2(temp);
        temp = temp + 1;
    else
        m2(i) = 0;
```

```
    end
end

% Multiply the filled arrays to obtain the final signal (m)
m = m1 .* m2;
end
```



Experiment 14 (Time Scaling)

A discrete time signal x(n) is shown in figure.

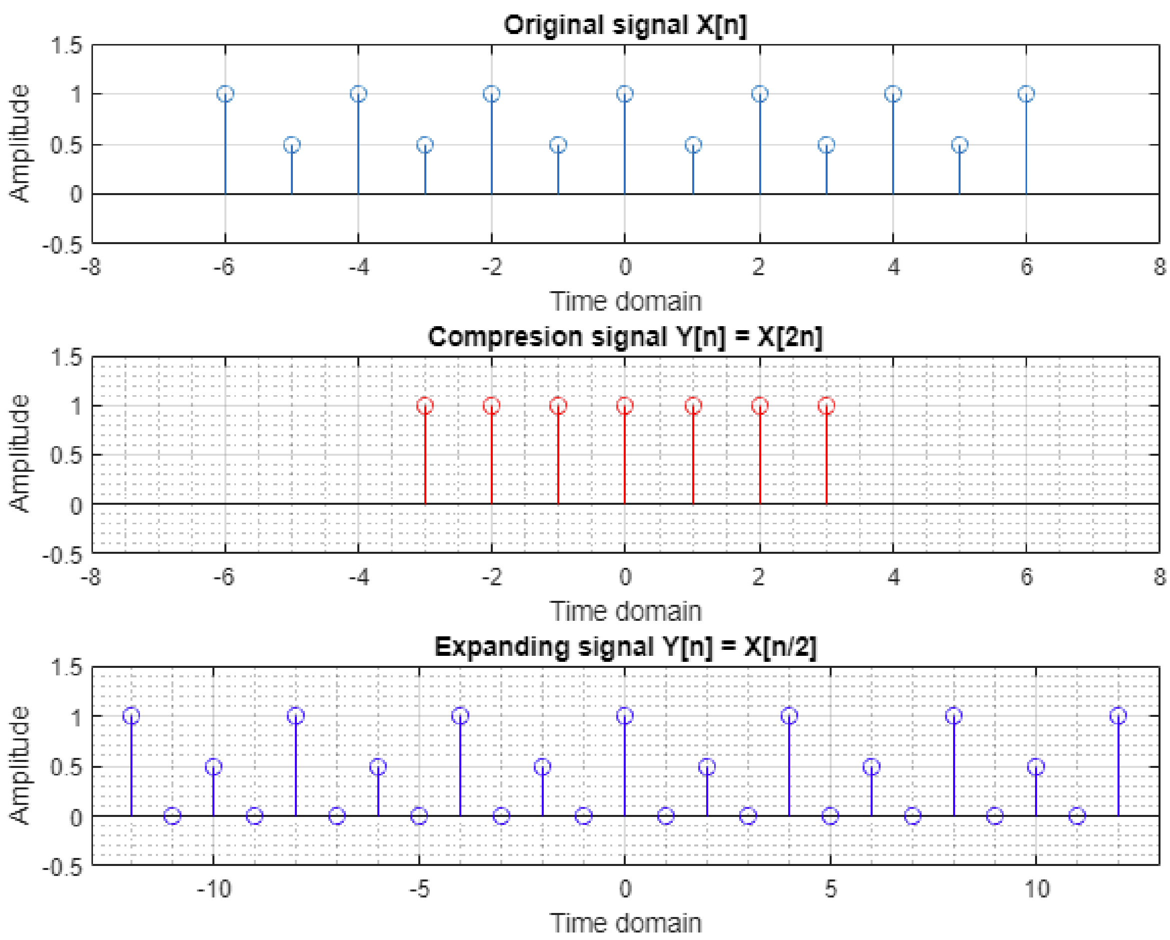


Sketch the signal x[n], the sketch y[n]=x[n/2] and y[n] = x[2n]

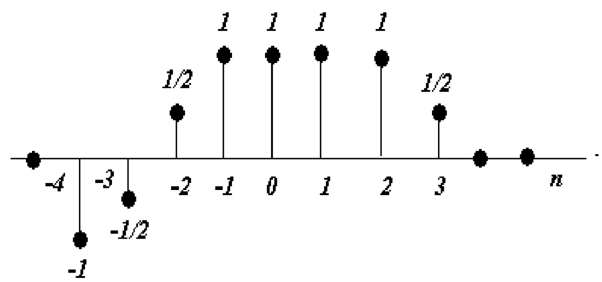
Solution:-

```
close all;
clear all;
clc;
n=-6:6;
y=[1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1];

index=1;
for i=1:length(n)
   if(rem(n(i),2)==0)
       x1(index)=n(i)./2;
       y1(index)=y(i);
       index=index+1;
   end;
end
subplot(3,1,1);
stem(n,y);
xlabel("Time domain");
ylabel("Amplitude");
grid on;
axis([-8 8 -0.5 1.5]);
title("Original signal X[n]");
subplot(3,1,2);
stem(x1,y1,'r');
xlabel("Time domain");
ylabel("Amplitude");
grid on;
grid minor;
axis([-8 8 -0.5 1.5]);
title("Compresion signal Y[n] = X[2n]");
index=1;
n2=-12:12;
for i=1:length(n2)
   x1(i)=n2(i);
   if(rem(n2(i),2)==0)
       y1(i)=y(index);
       index=index+1;
   else
       y1(i)=0;
```

```
    end
end
subplot(3,1,3);
stem(x1,y1,'b');
xlabel("Time domain");
ylabel("Amplitude");
grid on;
grid minor;
axis([-13 13 -0.5 1.5]);
title("Expanding signal Y[n] = X[n/2]");
```



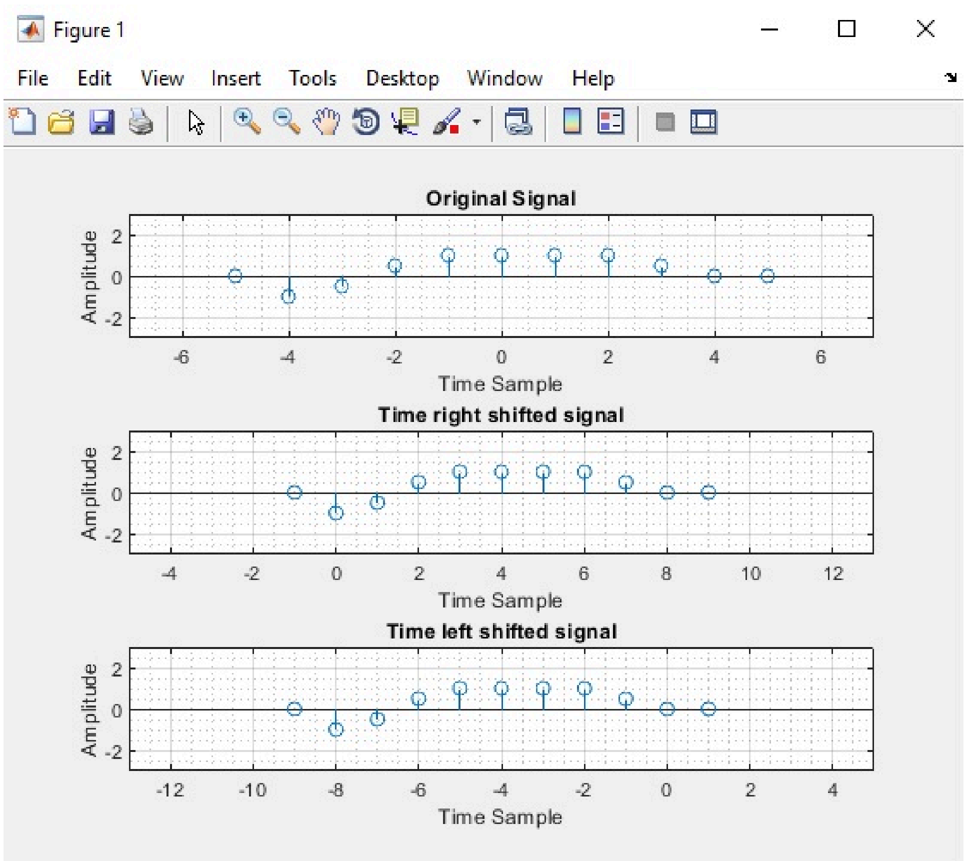
Experiment 15: A discrete time signal x(n) is shown in figure. Sketch the signal x[n], y[n]=x[n-4] and x[n+4], derived from x[n].



Solution:

```
clc;
clear;


n = -5:5;
x= [0 -1 -.5 .5 1 1 1 1 .5 0 0]
subplot(3,1,1);
stem (n,x);
xlabel('Time Sample');
ylabel('Amplitude');
title('Original Signal');
axis([-7 7 min(x)-2 max(x)+2]);
grid on;
grid minor;
```

```
m = n+4;
subplot(3,1,2);
stem (m,x);
xlabel('Time Sample');
ylabel('Amplitude');
title('Time right shifted signal');
axis([-7-2+4 7+2+4 min(x)-2 max(x)+2]);
grid on;
grid minor;

l = n-4;
subplot(3,1,3);
stem (l,x);
xlabel('Time Sample');

ylabel('Amplitude');
title('Time left shifted signal');
axis([-7-2-4 7+2-4 min(x)-2 max(x)+2]);
grid on;
grid minor;
```



Experiment 16: Find the even and odd components of the discrete-time signal x(n), where,

$$x[n] = \{5, 6, \underset{\uparrow}{3}, 4, 1\}$$

—-------

<u>Solution</u>

—-------

```
n = -2:2;
x = [5,6,3,4,1];

% Creating mirrored versions for negative indices
x_mirror = fliplr(x); %x_mirror = [1,4,3,5,5]
```

```
% even and odd components
xe = (x + x_mirror) / 2; %xe= ( x(n)+x(-n) )/2;
xo = (x - x_mirror) / 2; %xo= ( x(n)-x(-n) )/2;

% Plotting
subplot(4,1,1);
stem(n, x);
grid on;
axis([-3 3 -1 7]);
xlabel('n');
ylabel('Amplitude');
title('Original Signal');

subplot(4,1,2);
stem(n, x_mirror);
grid on;
axis([-3 3 -1 7]);
xlabel('n');
ylabel('Amplitude');
title('Reversed Signal');

subplot(4,1,3);
stem(n, xe, 'b');
grid on;
axis([-3 3 -1 6]);
xlabel('n');
ylabel('Amplitude');
title('Even Signal');

subplot(4,1,4);
stem(n, xo, 'b');
grid on;
axis([-3 3 -3 3]);
xlabel('n');
ylabel('Amplitude');
```

title('Odd Signal');

Experiment 17: A discrete time signal x(n) is given by

$$x[n] = \begin{cases} 1, & n = 1,2 \\ -1, & n = -1,2 \\ 0. & n = 0, \qquad |n| > 2 \;\; and \; |n| \leq 6 \end{cases}$$

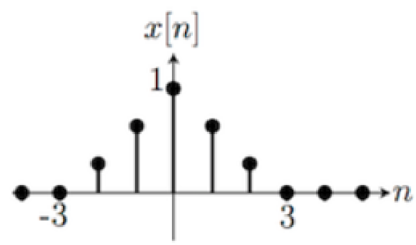
Sketch, y[n]=x[2n+3].

Solution

```
% Clear the workspace, close all figures, and clear the command
window
close all;
clear all;
clc;
% Define the original signal x[n] with time samples and
amplitudes
x1 = [-6:1:6]; % Time samples of original signal x[n]
y1 = [0 0 0 0 -1 -1 0 1 1 0 0 0 0]; % Corresponding amplitudes
of original signal x[n]
% Plot the original signal
subplot(3,1,1);
stem(x1, y1);
axis([-10 10 -2 2]); % Set axis limits for better visualization
xlabel("Time sample(n)");
ylabel("Amplitude");
title("Original Signal: x[n]");
% Generate a left-shifted version of the original signal x[n+3]
x2 = x1 - 3; % Shift the time samples of x1 by 3 units to the
left
y2 = y1; % Maintain the same amplitude values as the original
signal
% Plot the left-shifted signal
subplot(3,1,2);
stem(x2, y2);
axis([-10 10 -2 2]); % Same axis limits for comparison
xlabel("Time sample(n)");
```

```
ylabel("Amplitude");
title("Left Shifted Signal: x[n+3]");
% Define the scaling factor for amplitude compression
value = 2; % This factor will divide the amplitudes by 2
temp = 1; % Index counter for storing scaled samples
% Loop through the time samples of the shifted signal
for i = 1:length(x2)
  % if(rem(x2(i), 2)== 0)
   % Scale the time sample and store it in a new array (x3)
   x3(temp) = x2(i) / value;
   % Maintain the corresponding amplitude from the original
signal in y3
   y3(temp) = y2(i);
   % Increment the index counter for storing next scaled sample
   temp = temp + 1;
  % end
end
% Plot the final scaled and shifted signal
subplot(3,1,3);
stem(x3, y3);
axis([-10 10 -2 2]); % Same axis limits for comparison
xlabel("Time sample(n)");
ylabel("Amplitude");
title("Final Signal: x[2n+3]");
```

Experiment 17: Given the signal



Find y[n]=x[2n] and y[n]=x[n/2]

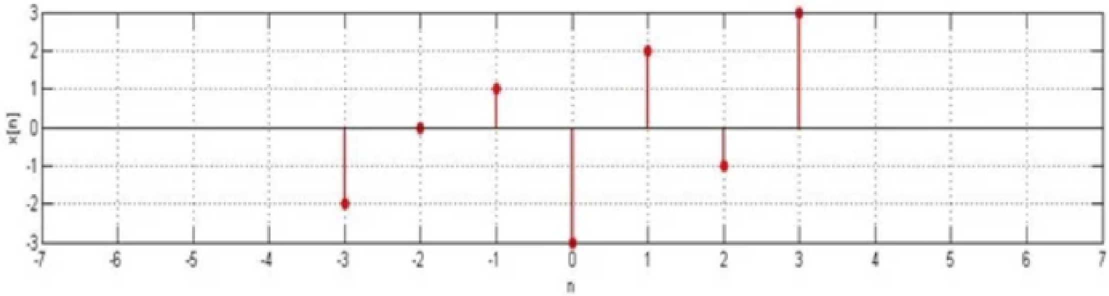
Solution:

% Close all open figures, clear workspace, and clear command window
close all;
clear all;
clc;

```
start_value = -4 %input('Enter the start value: ');%-6
end_value = 5 %input('Enter the end value: ');%6

n = start_value:end_value;

% Specify the compression factor
value = 2;

%y=input("Enter the values of signal = "); %[1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1]
y = [0 0 1/3 2/3 1 2/3 1/3 0 0 0];

% Initialize variables for compressed signal (Y[n] = X[2n])
index=1;
for i=1:length(n)
   if(rem(n(i),2)==0)
      x2(index)=n(i)./2;
      y2(index)=y(i);
      index=index+1;
   end
end


% Plot the compressed signal Y[n] = X[2n]
subplot(3,1,2);
stem(x2, y2, 'r');
xlabel("Time domain");
ylabel("Amplitude");
grid on;
grid minor;
axis([-8 8 -0.5 1.5]);
title("Compressed signal Y[n] = X[2n]");


% Initialize variables for expanded signal (Y[n] = X[n/2])
index=1;

n2=(2*start_value):(2*end_value);

for i=1:length(n2)
   x1(i)=n2(i);
```

```
    if(rem(n2(i),value)==0)
        y1(i)=y(index);
        index=index+1;
    else
        y1(i)=0;
    end
end

subplot(3,1,1);
stem(n,y,'r');
xlabel("Time");
ylabel("Amplitude");
grid on;
grid minor;
axis([(start_value-1) (end_value+1) -2 2]);
title("Original signal Y[n]=X[n]");


subplot(3,1,3);
stem(x1,y1,'b');
xlabel("Time");
ylabel("Amplitude");
grid on;
grid minor;
axis([(2*start_value-1) (2*end_value+1) -2 2]);
title("Expanded signal Y[n]=X[n/2]");
```

Experiment 18: Given the signal



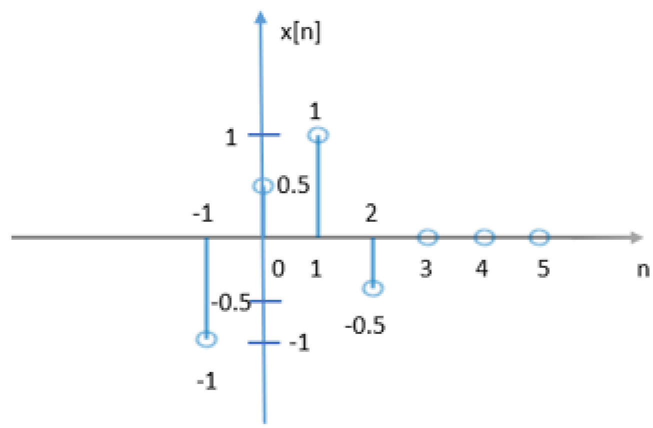
Find y[n]=x[n-3] and z[n]=x[n+2]

<u>Solution</u>:

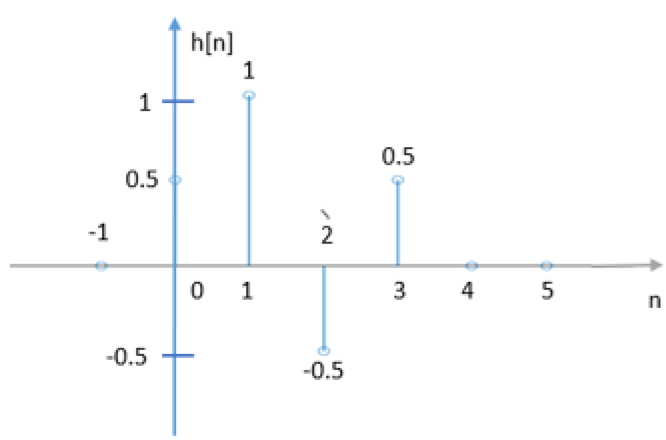
```
% Clear the command window and workspace
clc;
clear;
% Define the discrete time index
n = -3:3;
% Define the original signal X[n]
x = [-2 0 1 -3 2 -1 3];
% Plot the original signal X[n]
subplot(3,1,1);
stem(n, x);
xlabel('Time Sample');
ylabel('Amplitude');
title('Original Signal X[n]');
axis([-8 8 -4 4]);
grid on;
grid minor;
% Right-shift the signal by 3 units (Y[n] = X[n-3])
m = n + 3;
subplot(3,1,2);
stem(m, x, 'b');
xlabel('Time Sample');
ylabel('Amplitude');
title('Y[n] = X[n-3]');
axis([-8 8 -4 4]);
grid on;
grid minor;
% Left-shift the signal by 2 units (Z[n] = X[n+2])
l = n - 2;
subplot(3,1,3);
```

```
stem(l, x, 'r');
xlabel('Time Sample');
ylabel('Amplitude');
title('Z[n] = X[n+2]');
axis([-8 8 -4 4]);
grid on;
grid minor;
```

Experiment 19: The input x[n] of a LTI system,



The impulse response of the system:



Find out y[n].

Code:

Convolution.m

```
clc;
clear all;
close all;

x1=[-1 0 1 2];
y1=[-1 0.5 1 -0.5];
x2=[0 1 2 3 ];
h=[0.5 1 -0.5 0.5];

[n y]=func_convalution(x1,y1,x2,h);

subplot(3,1,1);
stem(x1,y1);
xlabel('X1');
ylabel('Y1');
title("Given Signal");

subplot(3,1,2);
stem(x2,h);
xlabel('x2');
ylabel('h');
title("Impulse Response");

subplot(3,1,3);
stem(n,y);
xlabel('n');
ylabel('y');
title("Convalution Sum");
```

func_convaluation.m:

```
function[n y]=func_convalution(x1,y1,x2,h)
m1=min(x1)+min(x2);
```

```
m2=max(x1)+max(x2);

n=m1:m2;
y=conv(y1,h); % build in function
```

Output: