Digital Signal Processing Lab

Name: Deep C. Patel

Roll No: 1401010

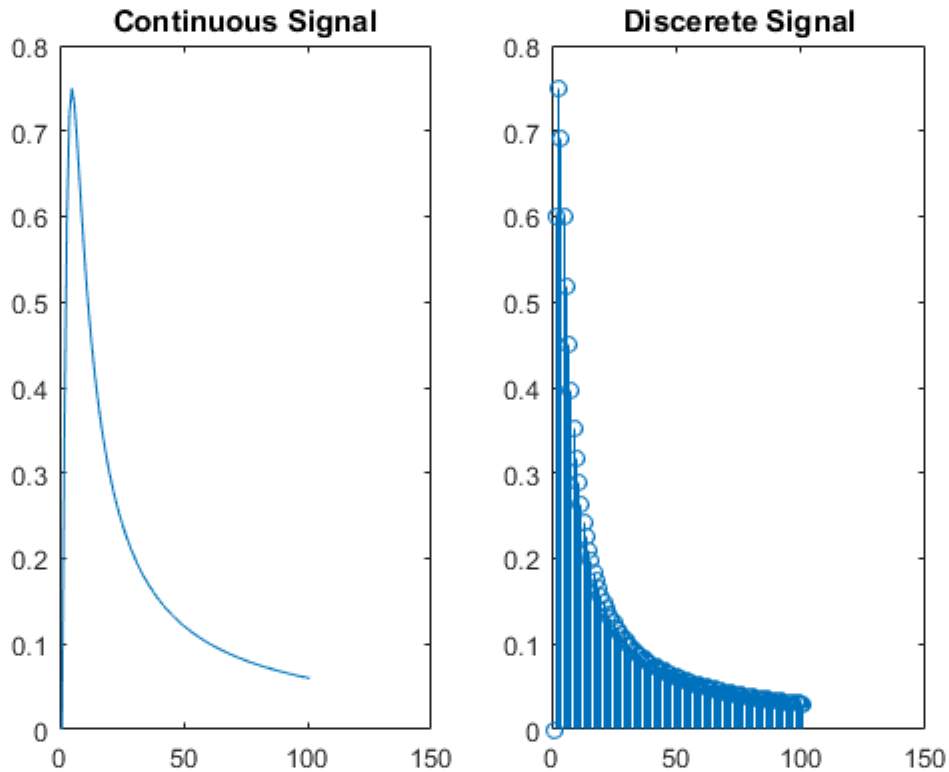
Lab Report

Lab Work:-

Lab – 1

1).

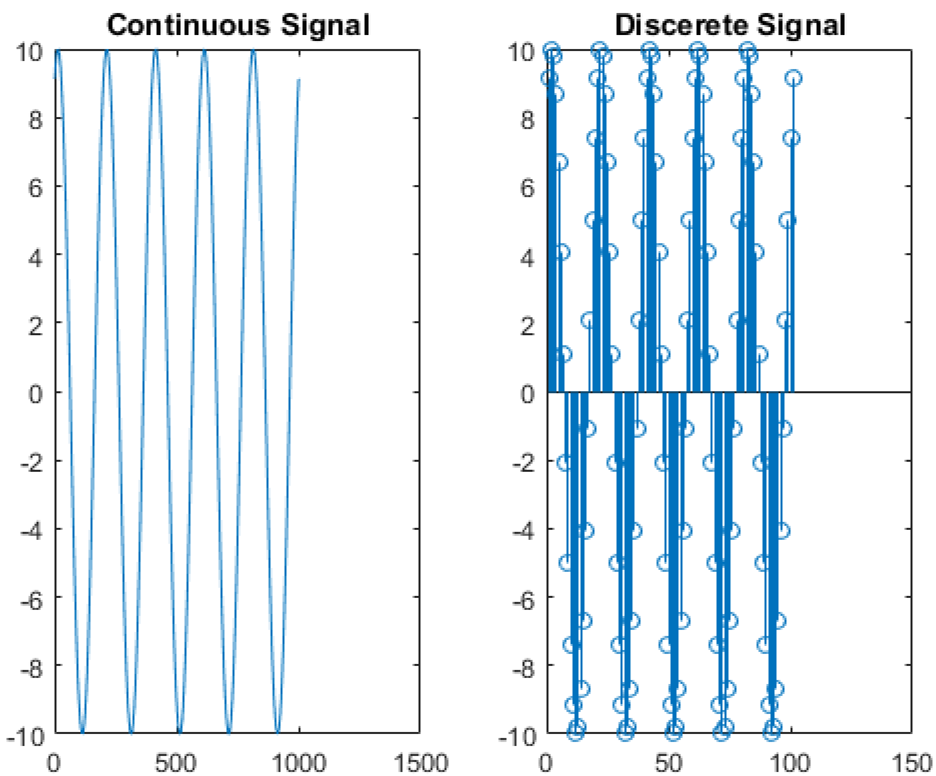
```
% Generate deterministic continuous time signal having equation
% x(t)= 3t/(4+t^2) and discrete time signal having equation x(n)= 3n/(4+n^2)

clc;
clear;

t=0:0.5:50;
n=0:1:100;

x_c=3*t./(4+t.^2);
x_d=3*n./(4+n.^2);

figure;
subplot(1,2,1);
plot(x_c);
title('Continuous Signal');

subplot(1,2,2);
stem(x_d);
title('Discerete Signal');
```



2).

```
% Plot the continuous and discrete time sinusoidal wave for given amplitude,
% frequency, phase and sampling frequency.

clc;
clear;

% Time specifications:
Fs = 1000;                    % samples per second or Sampling frequency
dt = 1/Fs;                    % seconds per sample
stop = 1;                     % seconds
t = 0:dt:stop;                % seconds
n = 0:1:100;                  % seconds
a = 10;                       % amplitude
phi = 20;                     % phase
Fc = 5;                       % Frequency in hertz

% Sine wave:

x_c = a*sin(2*pi*Fc*t + phi);
```

```
x_d = a*sin(2*pi*Fc*n/100 + phi);

figure;
subplot(1,2,1);
plot(x_c);
title('Continuous Signal');

subplot(1,2,2);
stem(x_d);
title('Discerete Signal');
```



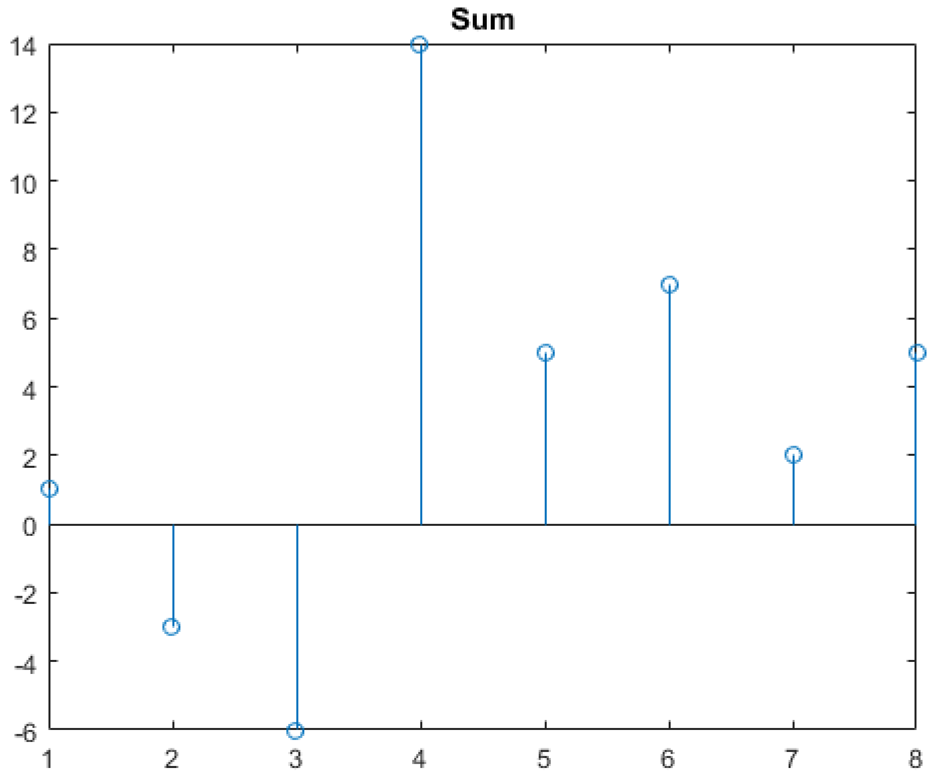
3).

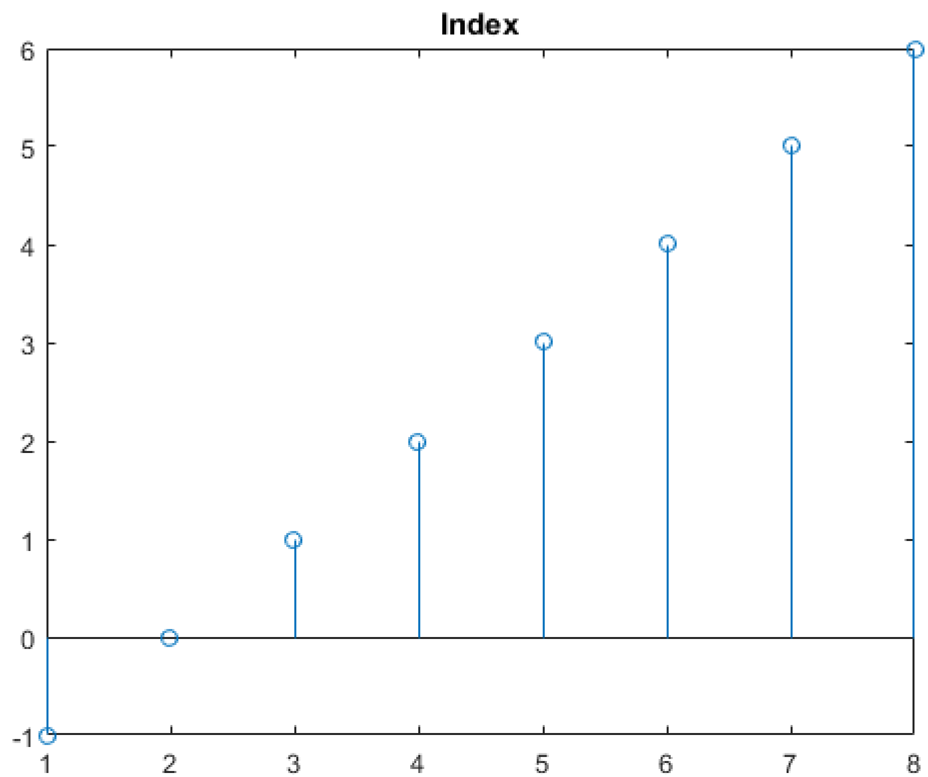
```
% Generate the function for signal addition. Add two sequences x 1 (n) ={1,-
1,2,5,1,5,-1} and x 2 (n) ={-2,-8,9,4,2,3,5}.

clc;
clear;

x1 = [1,-1,2,5,1,5,-1];
zero_ind_1 = 2;
x2 = [-2,-8,9,4,2,3,5];
zero_ind_2 = 1;
```

```
[x1, x2, sum, n] = sum_signal(x1, zero_ind_1, x2, zero_ind_2);

figure(3);


stem(sum);
title('Sum');

figure(4);

stem(n);
title('Index');
```

SUM_SIGNAL.M

```
%Sum of signals

function [x1, x2, sum, n] = sum_signal(x1, zero_ind_1, x2, zero_ind_2)

    len1 = length(x1);
    len2 = length(x2);

    if len1>len2
        x2 = [x2 zeros(1,(len1-len2))];
    elseif  len1<len2
        x1 = [x1 zeros(1,(len2-len1))];
    end

    if zero_ind_1==zero_ind_2

        sum = x1+x2;
        n = (1-zero_ind_1):1:(length(x1)-zero_ind_1);

    elseif zero_ind_1<zero_ind_2

        x1 = [zeros(1,(zero_ind_2-zero_ind_1)) x1];
        x2 = [x2 zeros(1,(zero_ind_2-zero_ind_1))];
        sum = x1+x2;
        n = (1-zero_ind_2):1:(length(x1)-zero_ind_2);

    else
        x1 = [x1 zeros(1,(zero_ind_1-zero_ind_2))];
        x2 = [zeros(1,(zero_ind_1-zero_ind_2)) x2];
        sum = x1+x2;
        n = (1-zero_ind_1):1:(length(x1)-zero_ind_1);
    end
end
```


Sum
14
12
10
8
6
4
2
0
-2
-4
-6
1 2 3 4 5 6 7 8


Index
6
5
4
3
2
1
0
-1
1 2 3 4 5 6 7 8

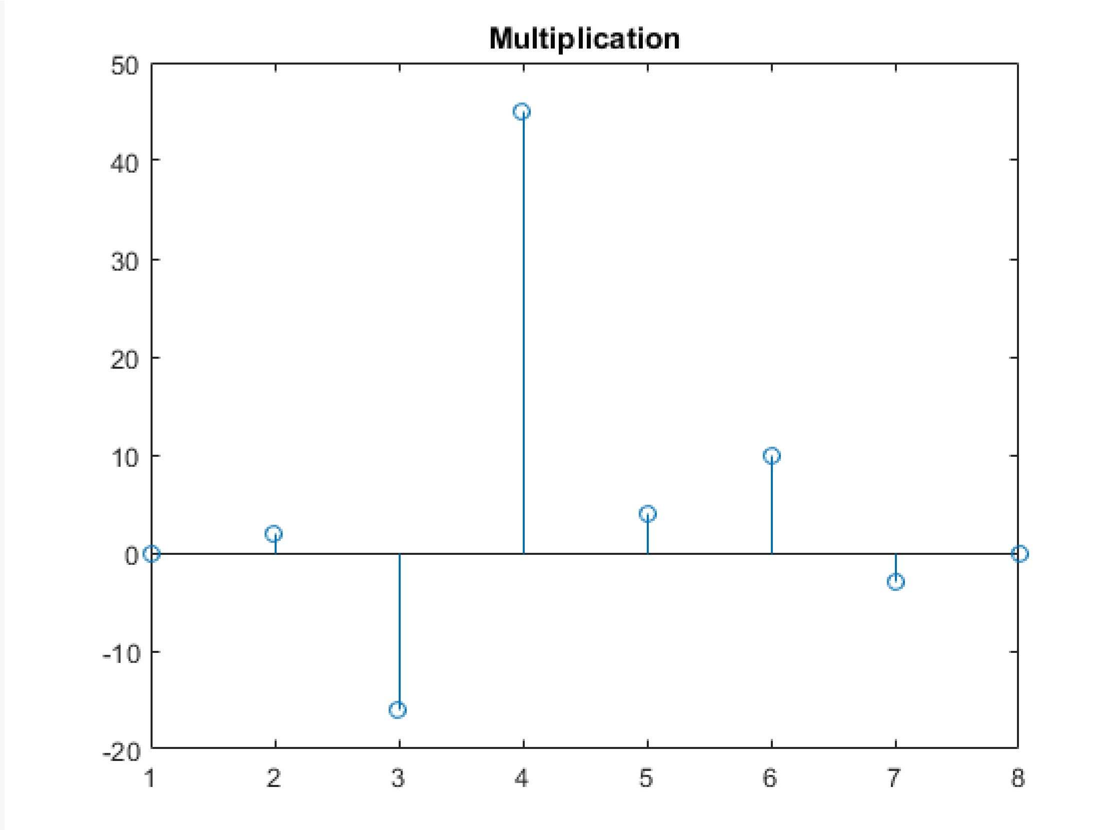
4).

```
% Generate the function for signal multiplication. For Two sequences x 1 (n) ={1,-1,2,5,1,5,-1} and x 2 (n) ={-2,-8,9,4,2,3,5}.

clc;
clear;

x1 = [1,-1,2,5,1,5,-1];
zero_ind_1 = 2;

x2 = [-2,-8,9,4,2,3,5];
zero_ind_2 = 1;

[x1, x2, mult, n] = multiply(x1, zero_ind_1, x2, zero_ind_2);

figure(3);


stem(mult);
title('Multiplication');

figure(4);

stem(n);
title('Index');
```

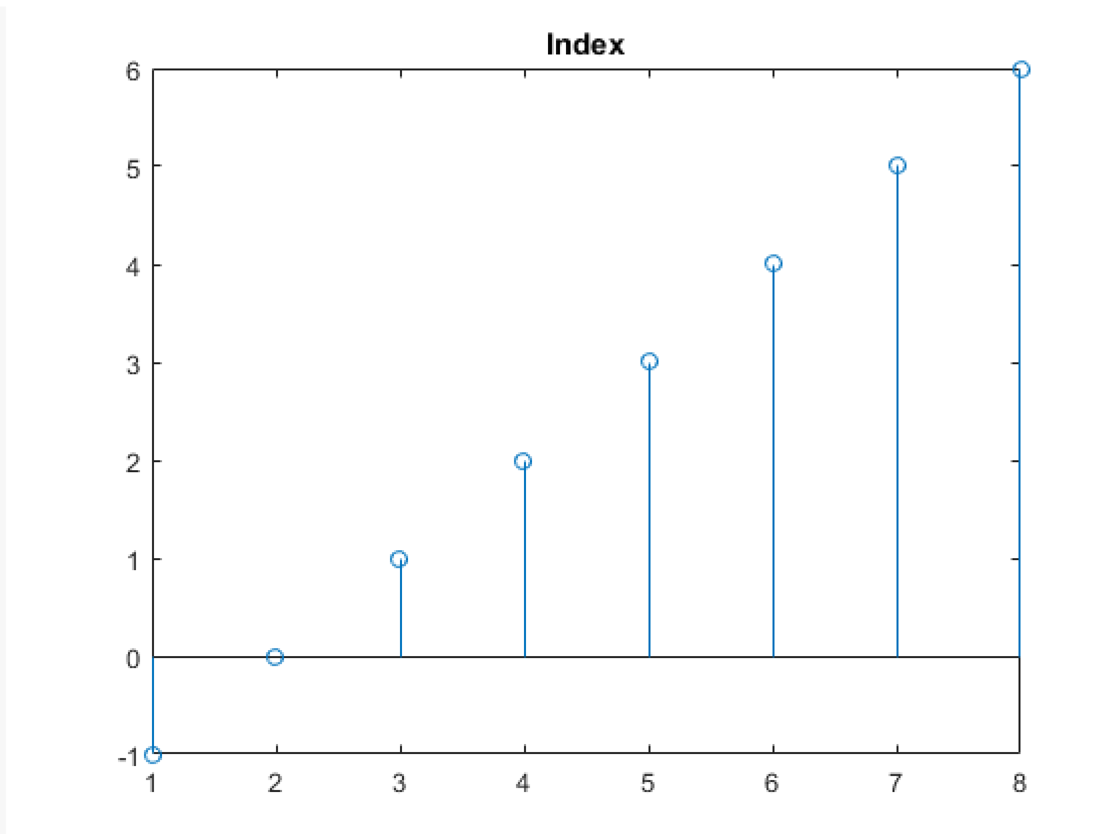
MULTIPLY.M

```
% Signal Multiplication

function [x1, x2, mult, n] = multiply(x1, zero_ind_1, x2, zero_ind_2)

    len1 = length(x1);
    len2 = length(x2);

    if len1>len2
        x2 = [x2 zeros(1,(len1-len2))];
    elseif  len1<len2
        x1 = [x1 zeros(1,(len2-len1))];
    end

    if zero_ind_1==zero_ind_2
        mult = x1.*x2;
        n = (1-zero_ind_1):1:(length(x1)-zero_ind_1);
    elseif zero_ind_1<zero_ind_2
        x1 = [zeros(1,(zero_ind_2-zero_ind_1)) x1];
        x2 = [x2 zeros(1,(zero_ind_2-zero_ind_1))];
        mult = x1.*x2;
        n = (1-zero_ind_2):1:(length(x2)-zero_ind_2);
```

```
    else
        x1 = [x1 zeros(1,(zero_ind_1-zero_ind_2))];
        x2 = [zeros(1,(zero_ind_1-zero_ind_2)) x2];
        mult = x1.*x2;
        n = (1-zero_ind_1):1:(length(x1)-zero_ind_1);
    end

end
```



Multiplication
50
40
30
20
10
0
-10
-20
1
2
3
4
5
6
7
8



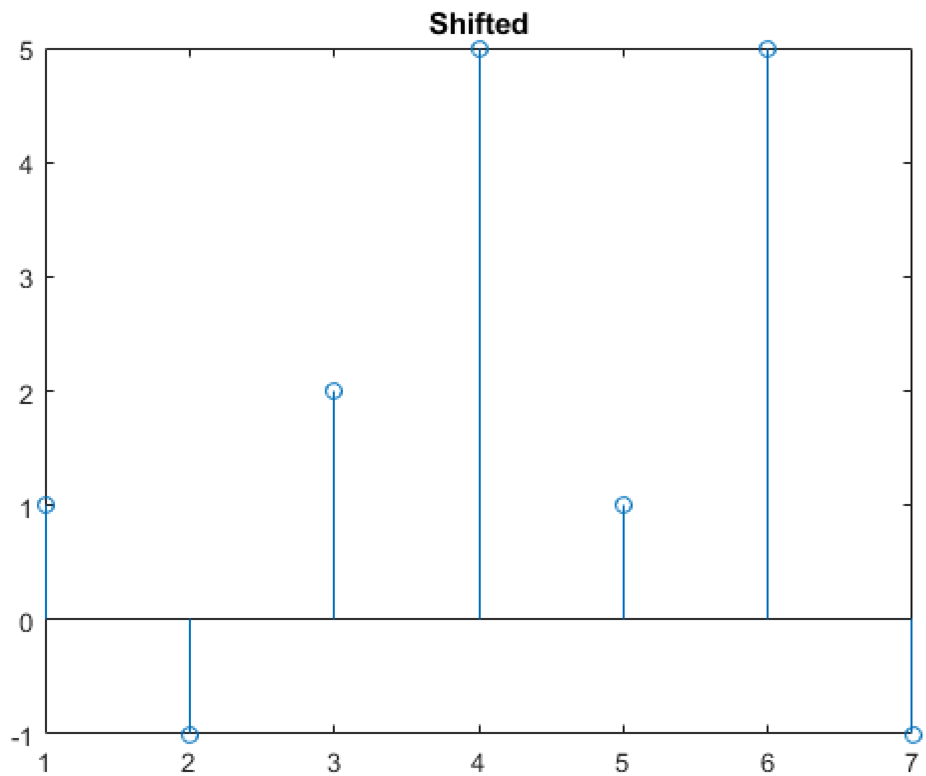
Index
6
5
4
3
2
1
0
-1
1
2
3
4
5
6
7
8

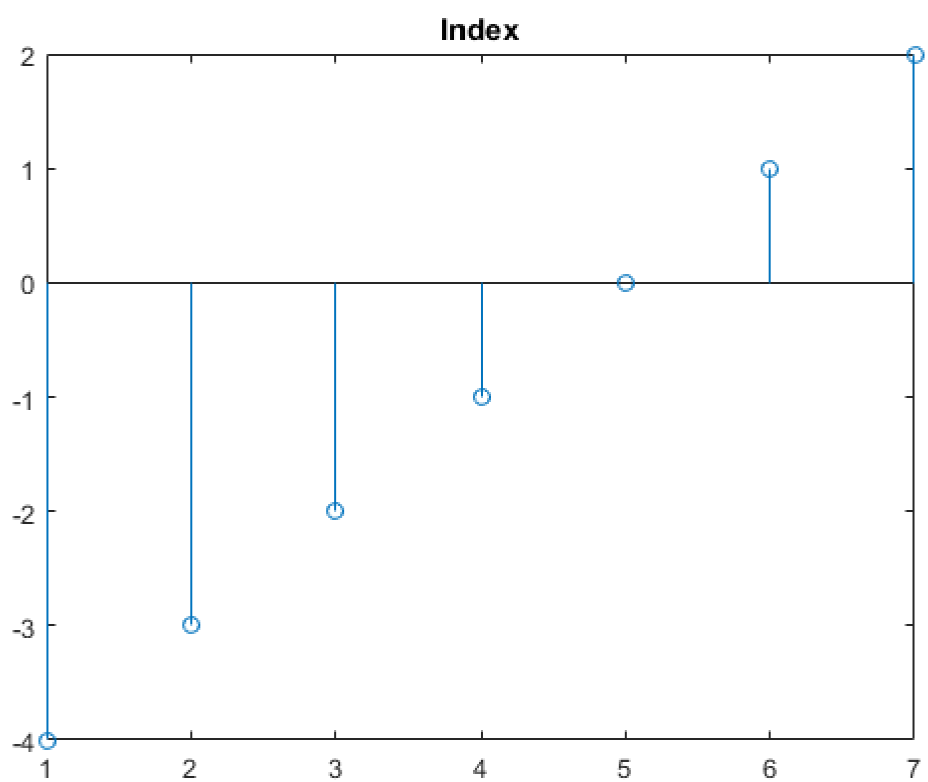
5).

```
% Generate the function for timing shifting. For sequences x(n) ={1,-1,2,5,1,5,-1}.

clc;
clear;

x =[1,-1,2,5,1,5,-1];
index_zero = 2;
sample = 3;

[n,Y] = shift(x,index_zero,sample,'advance');

figure(3);


stem(Y);
title('Shifted');

figure(4);

stem(n);
title('Index');
```

SHIFT.M

```
% Time Shift Function

function [n,Y] = shift(x,index_zero,sample,shifttype)

    Y = x;

    if strcmp(shifttype,'advance')==1
       n = (1-index_zero-sample):1:(length(Y)-index_zero-sample);
    elseif strcmp(shifttype,'delay')==1
       n = (1-index_zero+sample):1:(length(Y)-index_zero+sample);
    else
       disp('Error in Shift Type')
    end

end
```


Shifted
5
4
3
2
1
0
-1
1
2
3
4
5
6
7

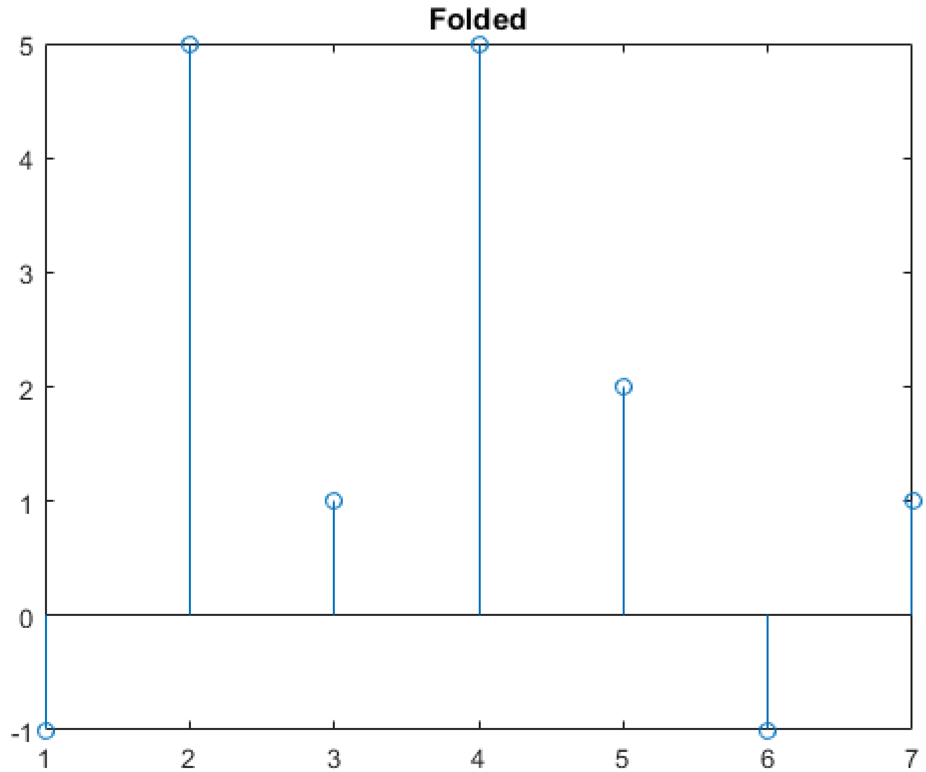

Index
2
1
0
-1
-2
-3
-4
1
2
3
4
5
6
7

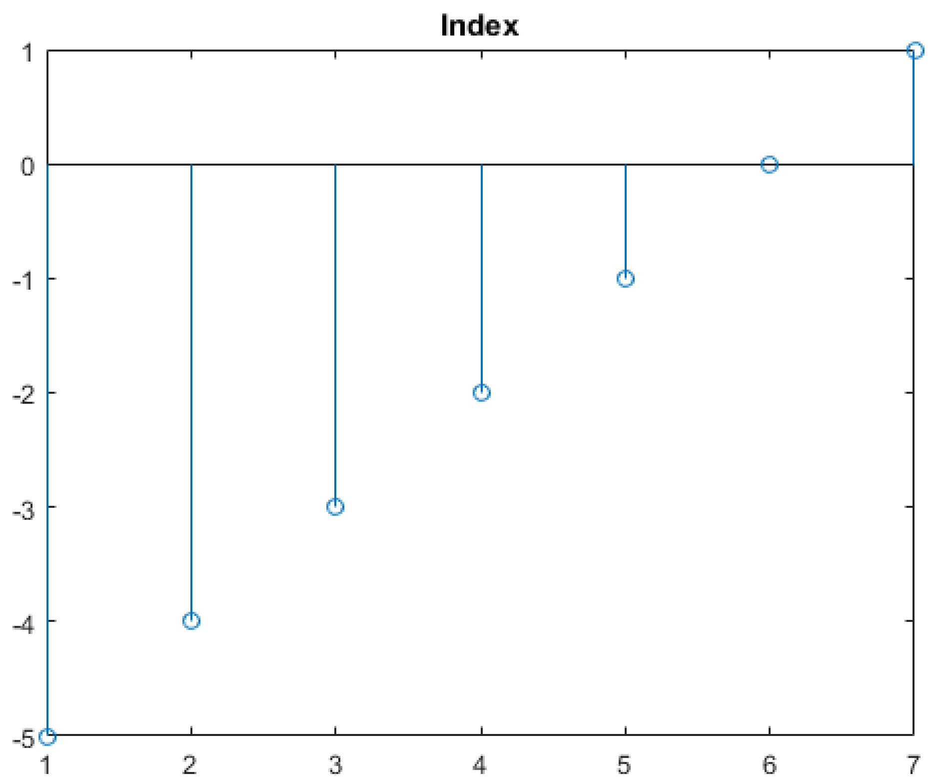
6).

```matlab
% Generate the function for signal folding. Fold the sequence x(n) ={1,-1,2,5,1,5,-1}.

clc;
clear;

x =[1,-1,2,5,1,5,-1];
index_zero = 2;
sample = 3;

[n,Y] = signal_folding(x,index_zero);

figure(3);


stem(Y);
title('Folded');

figure(4);

stem(n);
title('Index');
```

SIGNAL_FOLDING.M

```matlab
function [n,Y] = signal_folding(x,index_zero)

    Y = fliplr(x);
    index_zero = length(x)-index_zero+1;
    n = (1-index_zero):1:(length(x)-index_zero);

end
```


Folded
5
4
3
2
1
0
-1
1
2
3
4
5
6
7

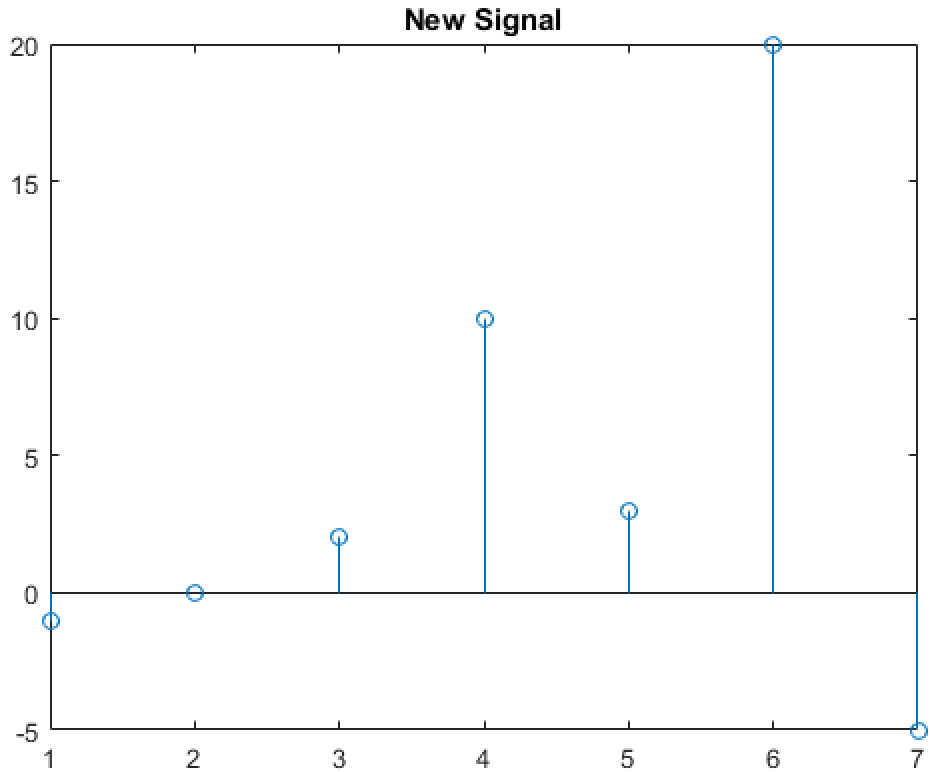

Index
1
0
-1
-2
-3
-4
-5
1
2
3
4
5
6
7

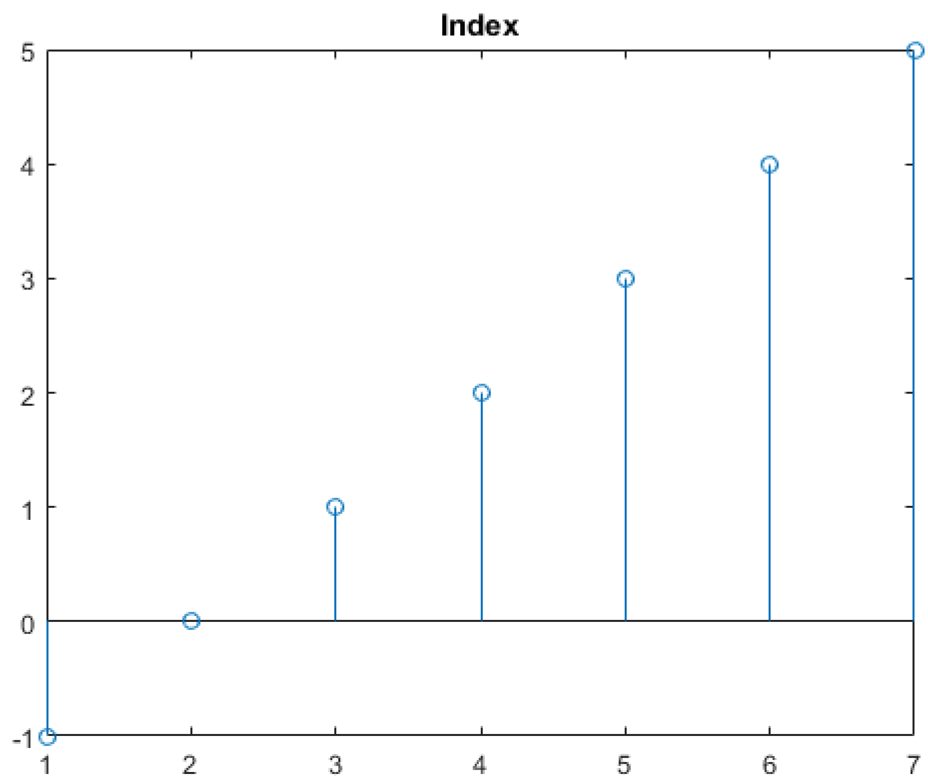
7).

```
% Generate the function for time multiplication. Use it for sequence x 1 (n) ={1,-
1,2,5,1,5,-1}.

clc;
clear;

x =[1,-1,2,5,1,5,-1];
index_zero = 2;
sample = 3;

[n,Y] = time_multiply(x,index_zero);

figure(3);


stem(Y);
title('New Signal');

figure(4);

stem(n);
title('Index');
```

TIME_MULTIPLY.M

```
function [n,Y] = time_multiply(x,index_zero)

    n = (1-index_zero):1:(length(x)-index_zero);
    Y = n.*x;

end
```


New Signal
20
15
10
5
0
-5
1
2
3
4
5
6
7

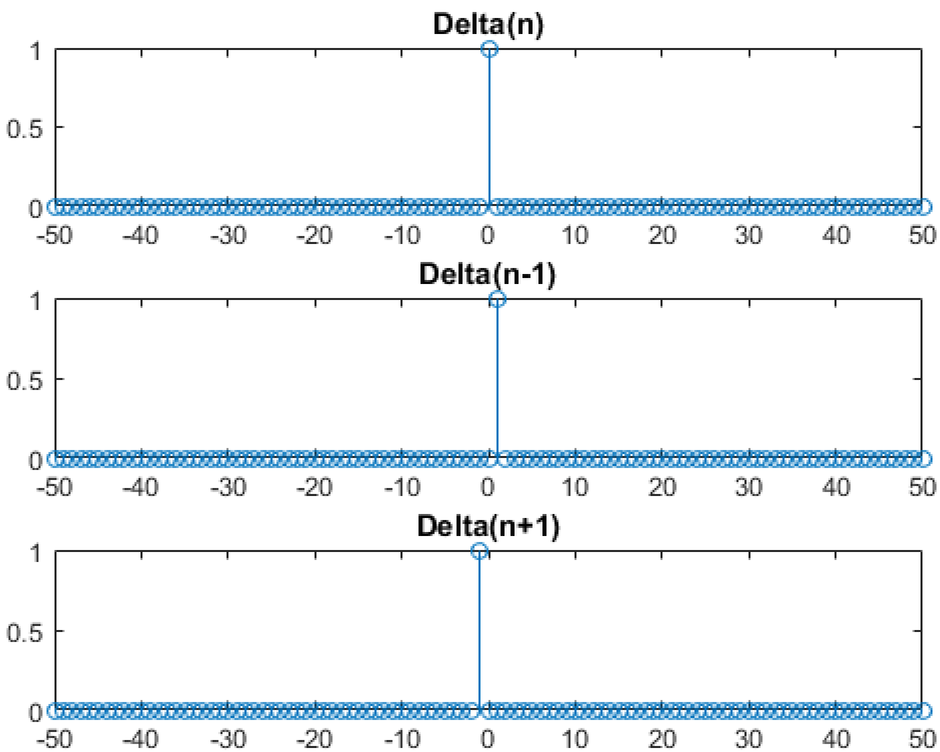

Index
5
4
3
2
1
0
-1
1
2
3
4
5
6
7

8).

```
% Generate function for unit step signal delta(n) Also plot delta(n-1) and delta(n+1).

clc;
clear;

n = -50:1:50;

figure;

x=delta(n);
x1=delta(n-ones(1,length(n)));
x2=delta(n+ones(1,length(n)));

subplot(3,1,1);
stem(n,x);
title('Delta(n)');

subplot(3,1,2);
stem(n,x1);
title('Delta(n-1)');

subplot(3,1,3);
stem(n,x2);
title('Delta(n+1)');
```

DELTA.M

```
% Delta Function

function u = delta(v)

    u = length(v);

    for i = 1:u

        if v(i)==0
            u(i)=1;
        else
            u(i)=0;
        end
    end
end
```



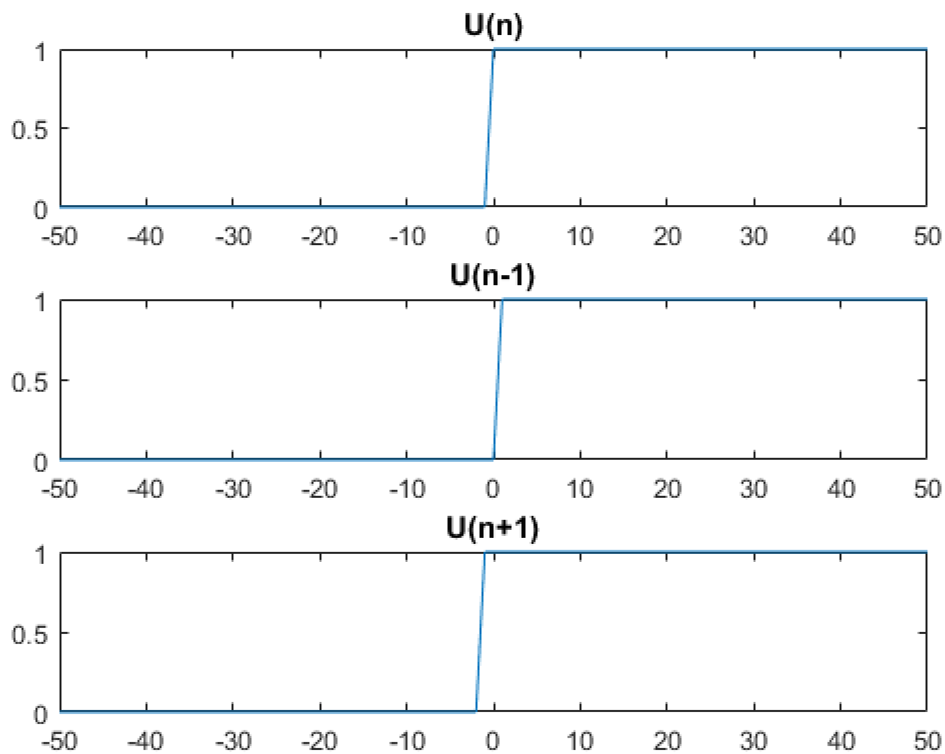
9).

```
% Generate function for unit step signal u(n) Also plot u(n-1) and u(n+1).

clc;
clear;

n = -50:1:50;

figure;

x=U(n);
x1=U(n-ones(1,length(n)));
x2=U(n+ones(1,length(n)));

subplot(3,1,1);
plot(n,x);
title('U(n)');

subplot(3,1,2);
plot(n,x1);
title('U(n-1)');
```

```
subplot(3,1,3);
plot(n,x2);
title('U(n+1)');
```

U.M

```
% Unit Step Function

function u = U(v)

    u = length(v);

    for i = 1:u

        if v(i)<0
            u(i)=0;
        else
            u(i)=1;
        end
    end
end
```

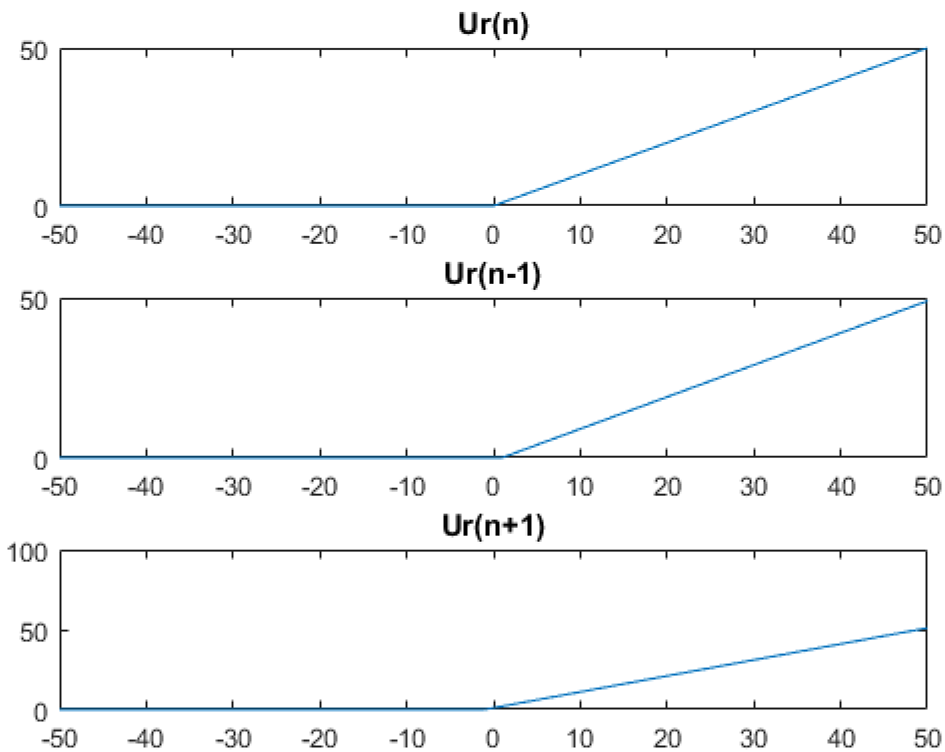


10).

```
% Generate function for unit ramp signal ur(n) Also plot ur(n + 1) and ur (n - 1).

clc;
clear;

n = -50:1:50;

figure;

x=Ur(n);
x1=Ur(n-ones(1,length(n)));
x2=Ur(n+ones(1,length(n)));

subplot(3,1,1);
plot(n,x);
title('Ur(n)');

subplot(3,1,2);
plot(n,x1);
title('Ur(n-1)');

subplot(3,1,3);
plot(n,x2);
title('Ur(n+1)');
```

UR.M

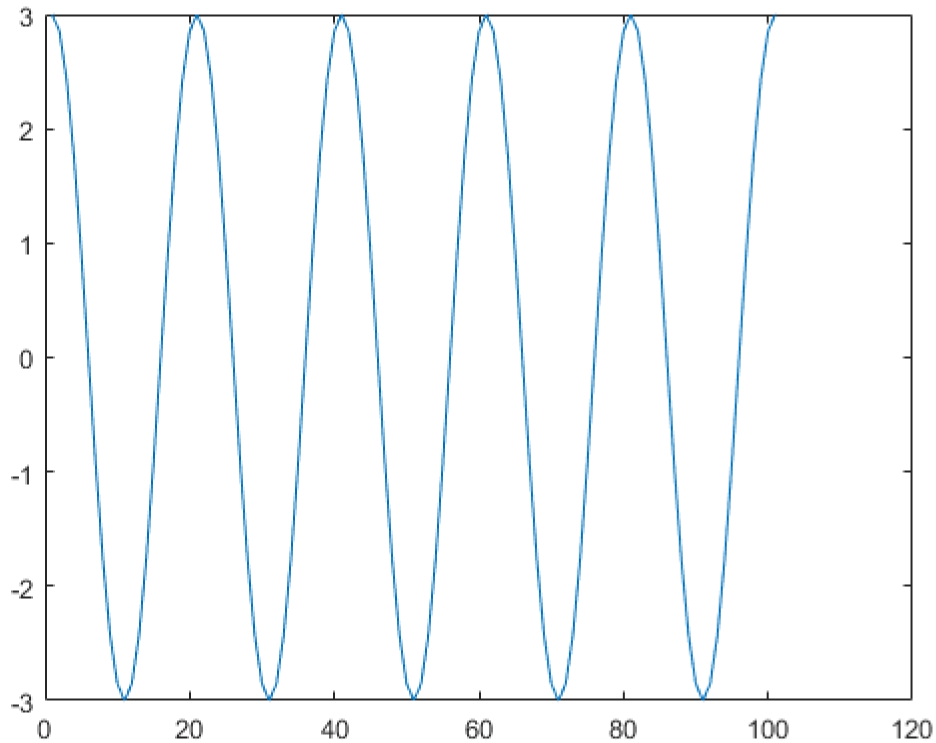
```
% Unit Ramp Function

function u = Ur(v)

    u = length(v);

    for i = 1:u

        if v(i)<0
            u(i)=0;
        else
            u(i)=v(i);
        end
    end
end
```



11).

```
% Find out the output of Accumulator if input x(n) = cos((2n*pi)/20)u(n).

clc;
clear;

stop = 100;                      % seconds
n = 0:1:stop;                   % seconds

% Cosine Wave:

x1 = 3*cos((pi*2/20)*n);

accumulate_x = sum(x1);

plot(x1);

fprintf('Accumulater output of all the elements of Signal X = %d \n', accumulate_x);
```

```
Accumulater output of all the elements of Signal X = 3.000000e+00
```



12).

```
% Perform the operation on x(n)
% x(n) = (1,-1,2,5,1,5,-1)
% (1). sum(-1,3,x) ;
% (2). product(-1,3,x) ;
% (3). energy(-infinite,infinite,x);

clc;
clear;

x = [1 -1 2 5 1 5 -1];
origin = 2;

sum_x = sum(x(1:5));

product_x = prod(x(1:5));

energy_x = x*x';

disp(['Sum of all the elements of Signal X = ', num2str(sum_x)]);
```

```
fprintf('Product of all the elements of Signal X = %d', product_x);
disp(sprintf('\nEnergy of the Signal X = %d', energy_x));
```

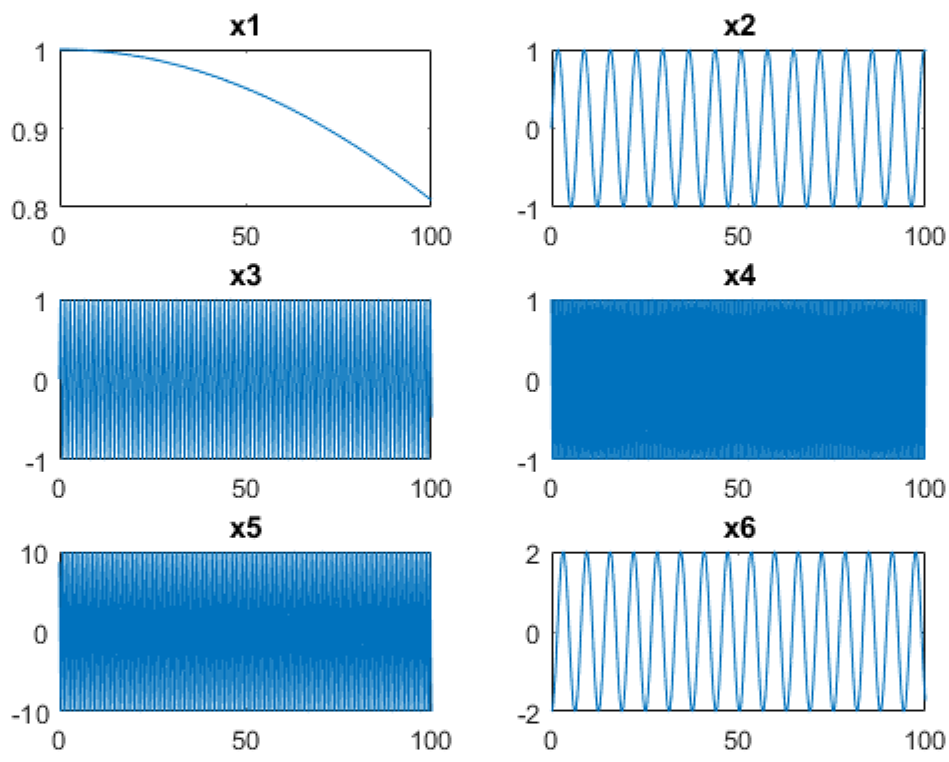
```
Sum of all the elements of Signal X = 8

Product of all the elements of Signal X = -10

Energy of the Signal X = 58
```

13).

```
% Plot all the given signals and comment on their output for periodicity writing
common
% MATLAB code.
% (1). x1 = cos((pi*0.002)*n);
% (2). x2 = sin((30*pi/105)*n);
% (3). x3 = sin(5*n);
% (4). x4 = cos((32*pi/10)*n);
% (5). x5 = 10*cos((7)*n + (pi/6));
% (6). x6 = 2*(cos(n-pi) + 1i*sin(n-pi));

clc;
clear;

% Time specifications:
Fs = 1000;                      % samples per second or Sampling frequency
dn = 1/Fs;                      % seconds per sample
stop = 100;                     % seconds
n = 0:dn:stop;                  % seconds

% Waves:

x1 = cos((pi*0.002)*n);
x2 = sin((30*pi/105)*n);
x3 = sin(5*n);
x4 = cos((32*pi/10)*n);
x5 = 10*cos((7)*n + (pi/6));
x6 = 2*(cos(n-pi) + 1i*sin(n-pi));

figure;

subplot(3,2,1);
plot(n, x1);
title('x1');

subplot(3,2,2);
```

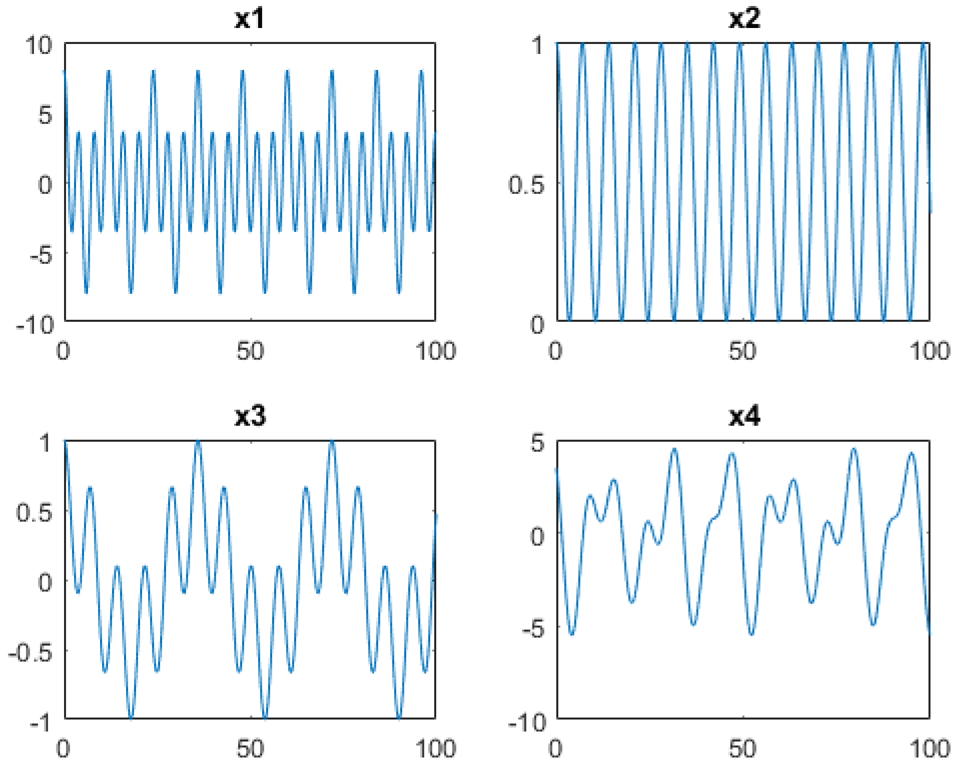
```
plot(n, x2);
title('x2');

subplot(3,2,3);
plot(n, x3);
title('x3');

subplot(3,2,4);
plot(n, x4);
title('x4');

subplot(3,2,5);
plot(n, x5);
title('x5');

subplot(3,2,6);
plot(n, x6);
title('x6');
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored
```



14).

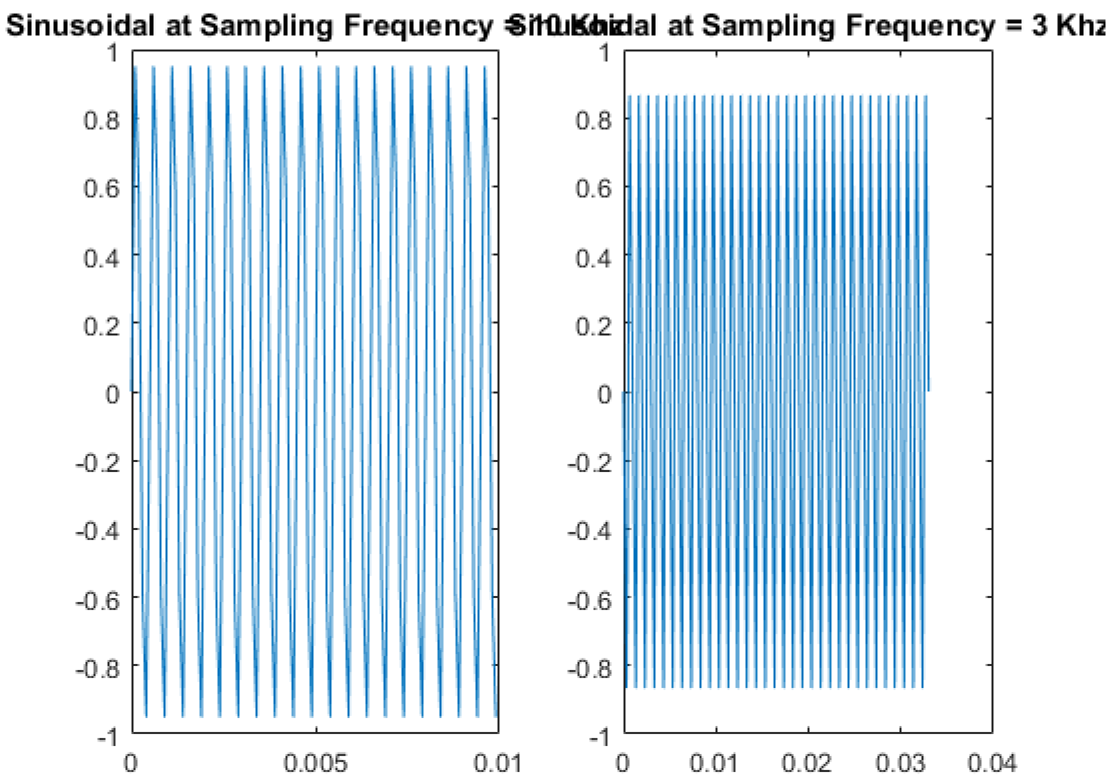
```
% Plot all the given signals and comment on their output for periodicity writing
common
% MATLAB code.
% (1). x1 = 3*cos((pi/6)*n) + 5*cos((3*pi/6)*n);
% (2). x2 = cos((pi/7)*n) .* cos((pi/7)*n);
% (3). x3 = cos((pi/6)*n) .* cos((pi/9)*n);
% (4). x4 = 2*cos((pi/4)*n) - sin((pi/6)*n) + 3*cos((pi/8)*n + (pi/3));

clc;
clear;

% Time specifications:
Fs = 1000;                    % samples per second or Sampling frequency
dn = 1/Fs;                    % seconds per sample
stop = 100;                   % seconds
n = 0:dn:stop;                % seconds

% Waves:

x1 = 3*cos((pi/6)*n) + 5*cos((3*pi/6)*n);
x2 = cos((pi/7)*n) .* cos((pi/7)*n);
x3 = cos((pi/6)*n) .* cos((pi/9)*n);
x4 = 2*cos((pi/4)*n) - sin((pi/6)*n) + 3*cos((pi/8)*n + (pi/3));

figure;

subplot(2,2,1);
plot(n, x1);
title('x1');

subplot(2,2,2);
plot(n, x2);
title('x2');

subplot(2,2,3);
plot(n, x3);
title('x3');

subplot(2,2,4);
plot(n, x4);
title('x4');
```



15).

```
% Sample the sinusoid x = sin(2 pi f t), where f = 2 kHz, and plot the sampled signals
over the
% continuous-time signal.
% - Let x1 be the signal sampled at 10 kHz.
% - Let x2 be the signal sampled at 3 kHz.
% Plot required waveforms and comment on the same by writing common MATLAB code.

clc;
clear;

% Time specifications:
Fs1 = 10*10^3;                 % samples per second or Sampling frequency 1
Fs2 = 3*10^3;                  % samples per second or Sampling frequency 2
dn1 = 1/Fs1;                   % seconds per sample
dn2 = 1/Fs2;                   % seconds per sample
stop = 1;                      % seconds
n1 = 0:dn1:stop;               % seconds
n2 = 0:dn2:stop;               % seconds
a = 1;                         % amplitude
```

```
Fc = 2*10^3;                    % Frequency in hertz

% Sine wave:

x1 = a*sin(2*pi*Fc*n1);

x2 = a*sin(2*pi*Fc*n2);

figure;
subplot(1,2,1);
plot(n1(1:100),x1(1:100));              %Plotting First 100 Samples only
title('Sinusoidal at Sampling Frequency = 10 Khz');

subplot(1,2,2);
plot(n2(1:100),x2(1:100));              %Plotting First 100 Samples only
title('Sinusoidal at Sampling Frequency = 3 Khz');
```



-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-