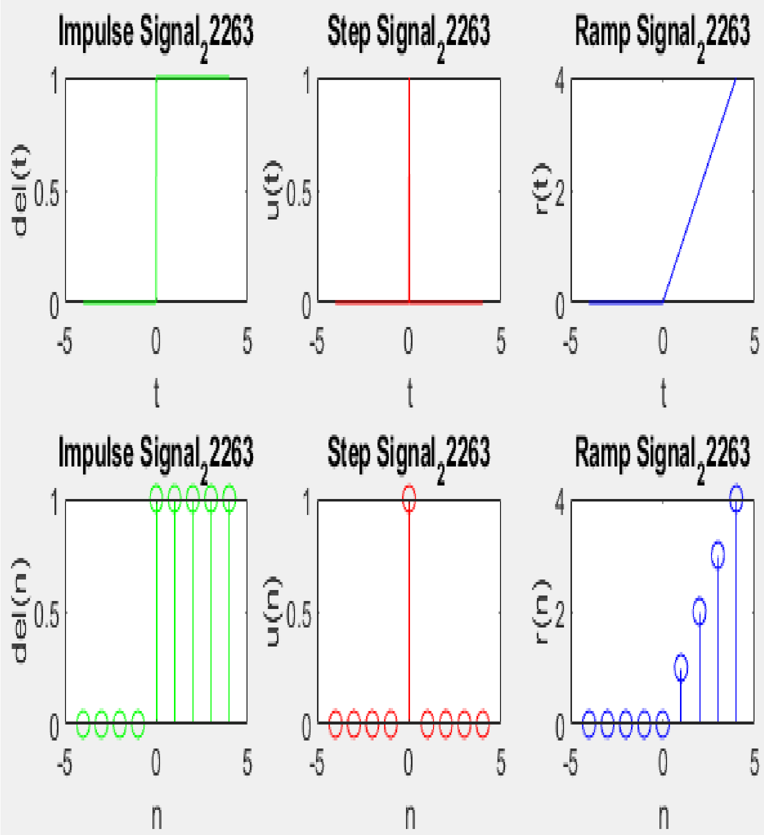


Aim:

To generate elementary signals such as unit impulse, unit step and unit ramp signal using user defined functions in MATLAB

• To generate composite signals such as exponential, triangular, rectangular in MATLAB.

• To perform the following operations on signals: 1. Linear combination 2. Time scaling and Time shifting

Questions:

1. Write a function to generate the following signals

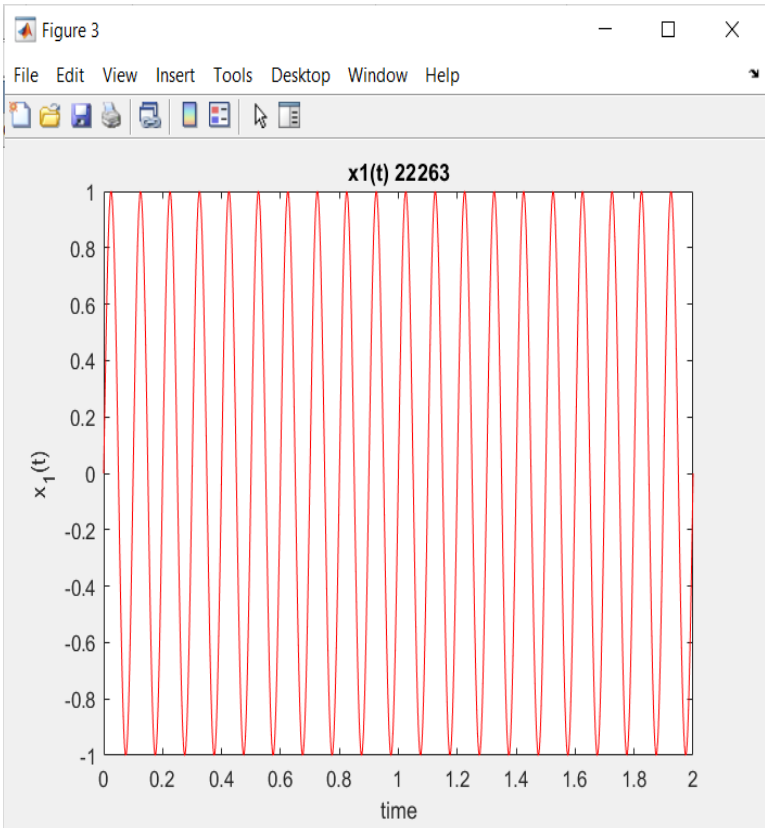
a) unit impulse signal $\delta[n]$ *and* $\delta(t)$

b) unit step signal $u[n]$ *and* $u(t)$

c) unit ramp signal $r[n]$ *and* $r(t)$

Code:

```
t_22263=-4:0.01:4;
n_22263=-4:1:4;


impulse_conti_22263=t_22263>=0;
step_conti_22263=t_22263==0;
ramp_conti_22263=t_22263.*(t_22263>=0);


impulse_dis=n_22263>=0;
step_dis=n_22263==0;
ramp_dis=n_22263.*(n_22263>=0);


subplot(3,3,1);
plot(t_22263,impulse_conti_22263,'g');
title('Impulse Signal_22263');
xlabel('t');
ylabel('del(t)');


subplot(3,3,2);
plot(t_22263,step_conti_22263,'r');
title('Step Signal_22263')
xlabel('t')
ylabel('u(t)')


subplot(3,3,3)
plot(t_22263,ramp_conti_22263,'b')
title('Ramp Signal_22263')
xlabel('t')
ylabel('r(t)')


subplot(3,3,4);
stem(n_22263,impulse_dis,'g');
```




```
title('Impulse Signal_22263');
xlabel('n');
ylabel('del(n)');


subplot(3,3,5);
stem(n_22263,step_dis,'r');
title('Step Signal_22263')
xlabel('n')
ylabel('u(n)')


subplot(3,3,6)
stem(n_22263,ramp_dis,'b')
title('Ramp Signal_22263')
xlabel('n')
ylabel('r(n)')
```

Result/Observation:

We can observe a unit step signal, ramp signal and an impulse signal in the discrete and continuous form in the output graph.

2.Create a vector t =0: 0.001 :2 , find x1(t)= sin($2\pi10$t) and x2(t) = cos($2\pi20$t).

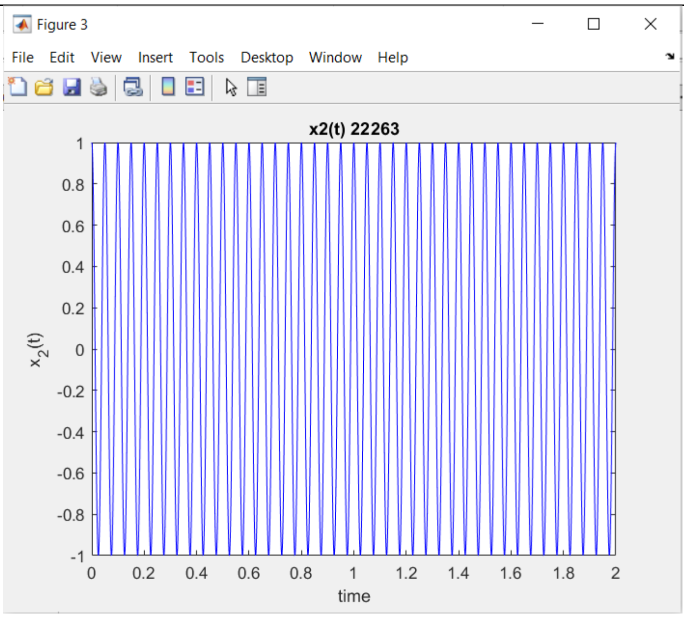
a) Plot the signals x1 and x2 versus t in two figure windows. (Hint :Use plot ,xlabel, ylabel)

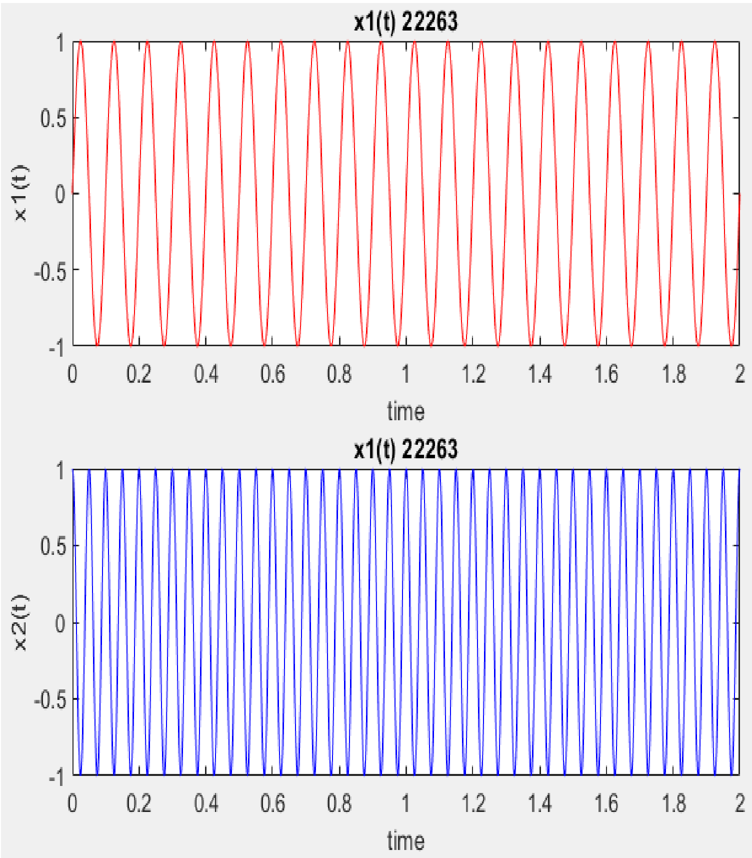
b) Plot the signals x1 and x2 versus t in the same figure window. (Hint: Use subplot, plot)

for part a.)

Code:

```
t_22263=0:0.001:2;


x1_t_22263=sin(2*pi*10*t_22263);
x2_t_22263=cos(2*pi*20*t_22263);


% for first figure window
plot(t_22263,x1_t_22263,'r')
xlabel('time')
ylabel('x_1(t)')
title('x1(t) 22263')


% for second figure window
plot(t_22263,x2_t_22263,'b')
xlabel('time')
ylabel('x_2(t)')
title('x2(t) 22263')
```





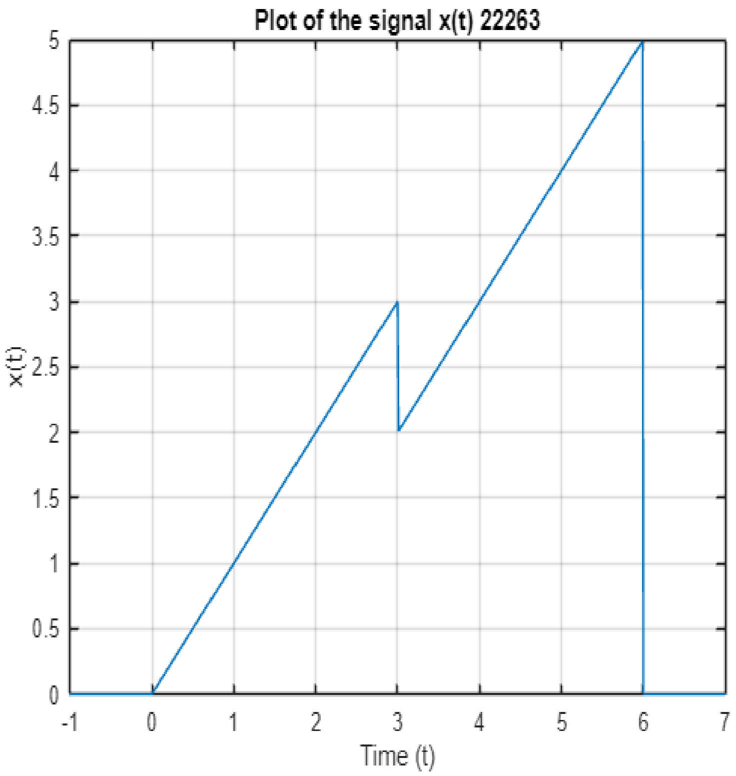
Part b)

```
t_22263=0:0.001:2;


x1_t_22263=sin(2*pi*10*t_22263);
x2_t_22263=cos(2*pi*20*t_22263);


% same window
subplot(2,1,1)
plot(t_22263,x1_t_22263,'r')
xlabel('time')
ylabel('x1(t)')
title('x1(t) 22263')


subplot(2,1,2)
plot(t_22263,x2_t_22263,'b')
xlabel('time')
ylabel('x2(t)')
title('x1(t) 22263')
```

Result/Observation:

We can observe in part a, the figures are plotted in two separate windows, but in part b we use subplot to plot them in the same window.

3.Write a program to generate the following signals in MATLAB



(a) $x(t)=\begin{cases} t & 0 \le t \le 3 \\ t-1 & 3 < t < 6 \\ 0 & Elsewhere \end{cases}$

(b) x[n]=u[n]-u[n-2] where u[n] is unit step signal

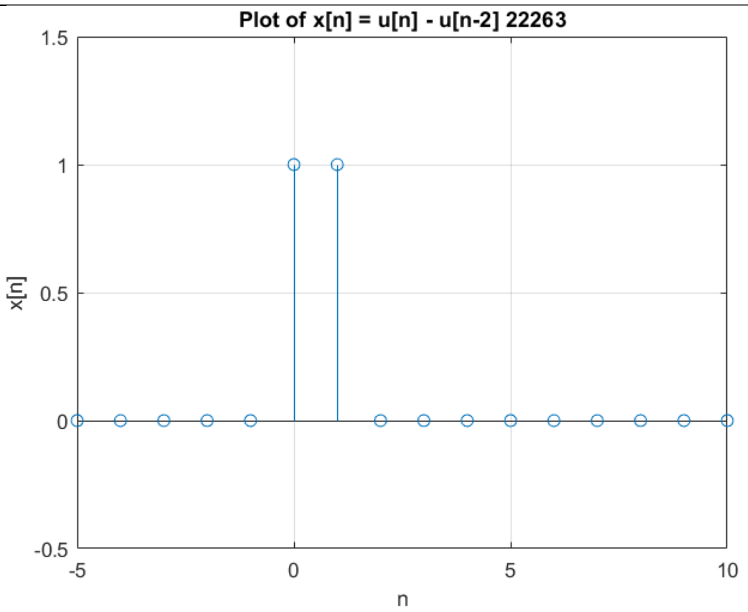
Code:

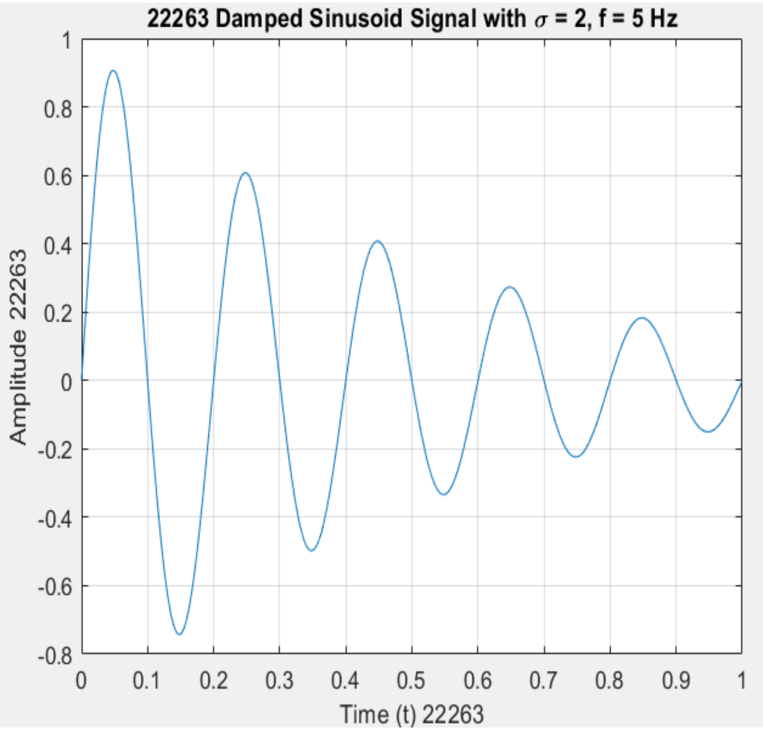
Part a:

```
t_22263 = -1:0.01:7;


% Initialize x(t) as a zero vector of the same size as t
x_22263 = zeros(size(t_22263));


% Apply the conditions for x(t)
% For 0 ≤ t ≤ 3
x_22263(t_22263 >= 0 & t_22263 <= 3) = t_22263(t_22263 >= 0 & t_22263 <= 3);


% For 3 < t < 6
x_22263(t_22263 > 3 & t_22263 < 6) = t_22263(t_22263 > 3 & t_22263 < 6) - 1;


% Plotting the signal
plot(t_22263, x_22263);
xlabel('Time (t)');
ylabel('x(t)');
title('Plot of the signal x(t) 22263');
grid on;
```

part b)

```
% Define the range for n
n_22263 = -5:10;


% Generate u[n]
u_22263 = n_22263 >= 0;  % This creates a logical array where the condition n >= 0 is true


% Generate u[n-2] by shifting u[n] two places to the right
% For MATLAB indexing, you need to handle the shift manually because MATLAB does not support negative indexing
```






```
u_shifted__22263 = [zeros(1, 2), u_22263(1:end-2)];  % Prepend(to add something to the beginning) two zeros for the shift


% Calculate x[n] = u[n] - u[n-2]
x_22263 = u_22263 - u_shifted__22263;


% Plot x[n]
stem(n_22263, x_22263);
xlabel('n');
ylabel('x[n]');
title('Plot of x[n] = u[n] - u[n-2] 22263');
axis([min(n_22263) max(n_22263) -0.5 1.5]); % We can use axis to adjust axis for better visualization
grid on;
```

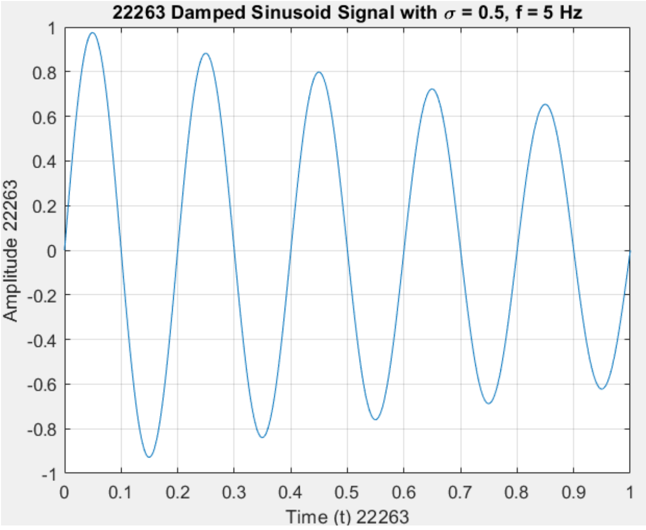
Result/Observation:

We can observe the output graphs x(t) and x(n) according to the given functions.

4.Write a user-defined function which generates an exponential damped sinusoid signal $x(t) = e^{-\sigma t}\sin(2\pi f t)$. Generate the signal for various values of σ and observe the change in the signal generated (Hint: Take σ, f, t as parameters of the function)

Code(Function):

```
function x_22263 = generateDampedSinusoid(sigma_22263, f_22263, t_22263)
    % generateDampedSinusoid generates an exponential damped sinusoid signal
    % Inputs:
    %   sigma - damping coefficient
    %   f - frequency of the sinusoid in Hz
    %   t - time vector

    % Calculate the signal
    x_22263 = exp(-sigma_22263 * t_22263) .* sin(2 * pi * f_22263 * t_22263);

    % Plot the signal
    plot(t_22263, x_22263);
    ++
    xlabel('Time (t) 22263');
```




```
    ylabel('Amplitude 22263');
    title(['22263 Damped Sinusoid Signal with
\sigma = ', num2str(sigma_22263), ', f = ',
num2str(f_22263), ' Hz']);
    % num2str is useful for labeling and titling
plots with numeric values.
    grid on;
end
```

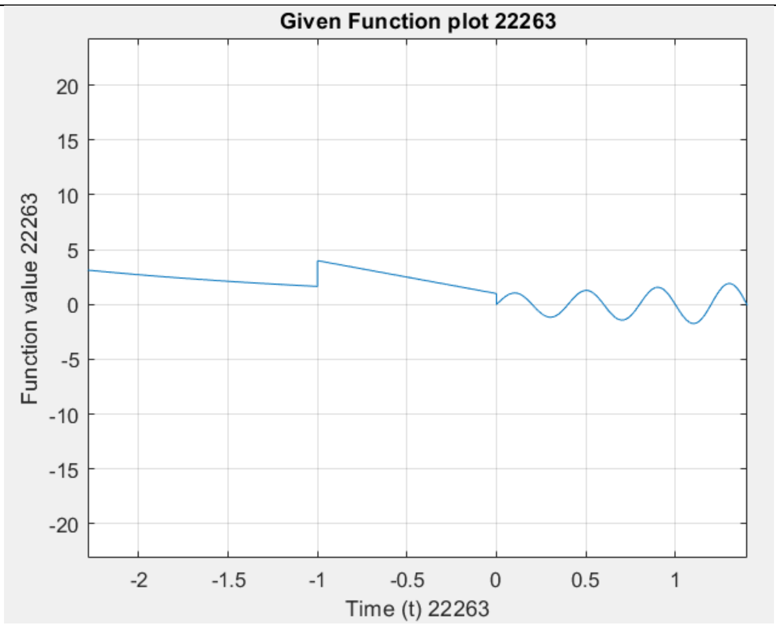
Code(main):

```
t_22263 = 0:0.001:1; % From 0 to 1 second with a
step of 1 ms

% Parameters for the signal
sigma1_22263 = 0.5;
f1_22263 = 5; % 5 Hz

figure;
generateDampedSinusoid(sigma1_22263, f1_22263,
t_22263);

% On trying different values of sigma and f
sigma2_22263 = 2;
f2_22263 = 5; % Keeping frequency same, increasing
damping

figure;
generateDampedSinusoid(sigma2_22263, f2_22263,
t_22263);
```

Result/Observation:

We can observe 2 examples of an exponential damped sinusoidal signal, also we can observe the change in the dampening on changing the values of σ.

5. Plot the signal given in Figure 1. (Hint: The first section in the figure is an exponentially decaying signal $e^{-0.5t}$, -3t+1 from -1 to 0 and last part is $e^{0.5t}\sin(5\pi t)$)



Code:

```
% Define the time vectors for each interval
t1_22263 = linspace(-10, -1, 1000); % From -10 to -1
t2_22263 = linspace(-1, 0, 1000);    % From -1 to 0
t3_22263 = linspace(0, 4, 1000);    % From 0 to 4

% Calculate the function values for each interval
y1_22263 = exp(-0.5 * t1_22263);
y2_22263 = -3 * t2_22263 + 1;
y3_22263 = exp(0.5 * t3_22263) .* sin(5 * pi * t3_22263);

t_22263=[t1_22263,t2_22263,t3_22263];
y_22263=[y1_22263,y2_22263,y3_22263];
figure;
% Plot each part of the function
plot(t_22263,y_22263);
xlabel('Time (t) 22263');
ylabel('Function value 22263');
title('Given Function plot 22263');
grid on;
```

Result/Observation:

We can observe the output graph to be as how the function was given.

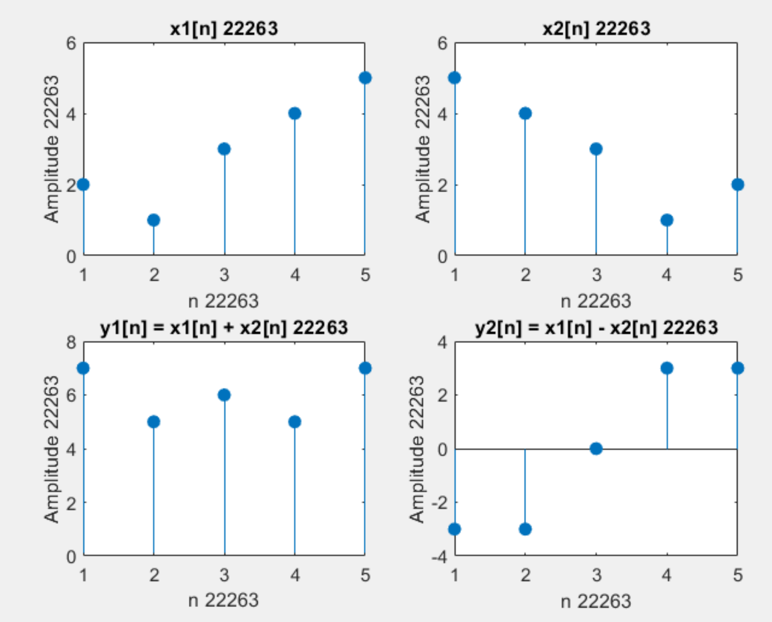
6. Write a function lincomp_RollNO which takes 4 inputs- a,b, $x1[n]$ and $x2[n]$ and generates the signal $x3[n] = a\,x1[n] + b\,x2[n]$ where a and b are scalars. Using the abovementioned function, generate the following signals $y1[n] = x1[n] + x2[n]$ and $y2[n] = x1[n] - x2[n]$ where $x1[n]$ = [2,1,3,4,5] and $x2[n]$ = [5,4,3,1,2]. Assume the first value to be the amplitude corresponding to n=0 for both the signals.

Code(Function):

```
function x3 = lincomp_RollNO(a, b, x1, x2)
    % Linear combination of two signals
    x3 = a .* x1 + b .* x2;
end
```

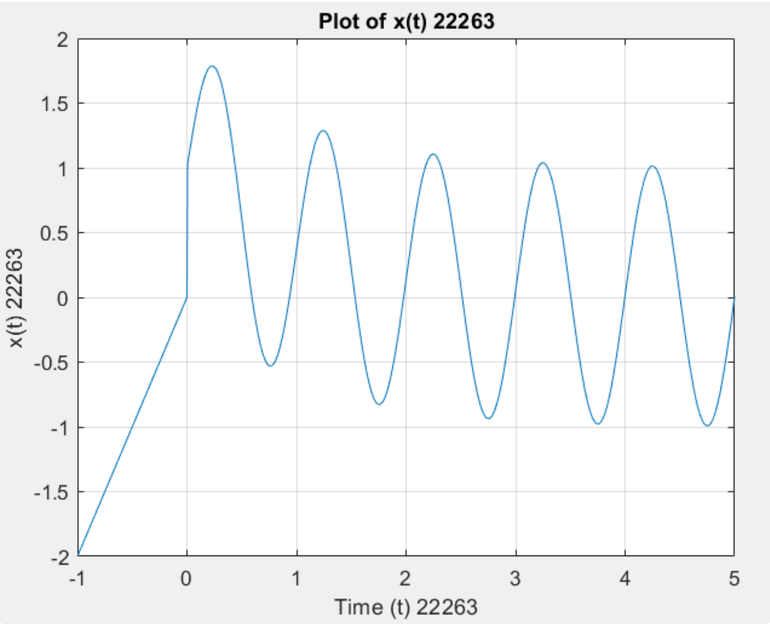
Code(main):

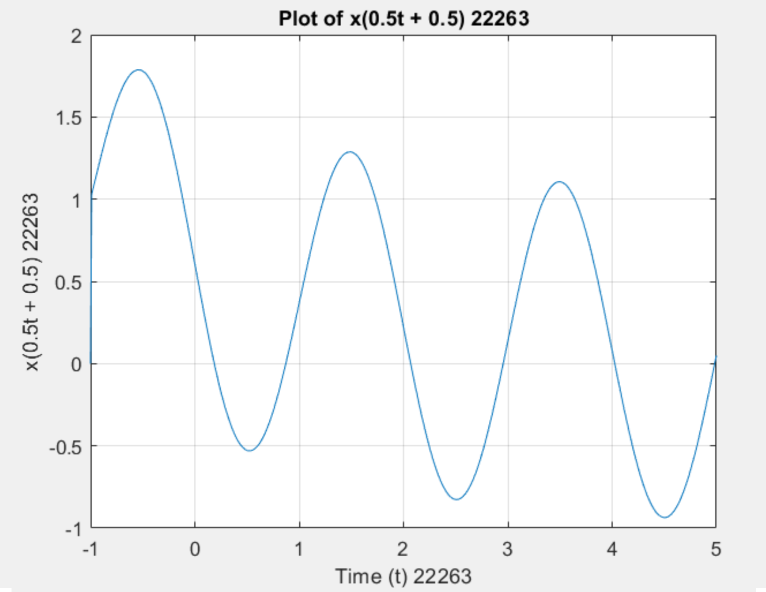
```
% Define the signals x1[n] and x2[n]
x1_22263 = [2, 1, 3, 4, 5];
x2_22263 = [5, 4, 3, 1, 2];

% Generate y1[n] = x1[n] + x2[n]
```

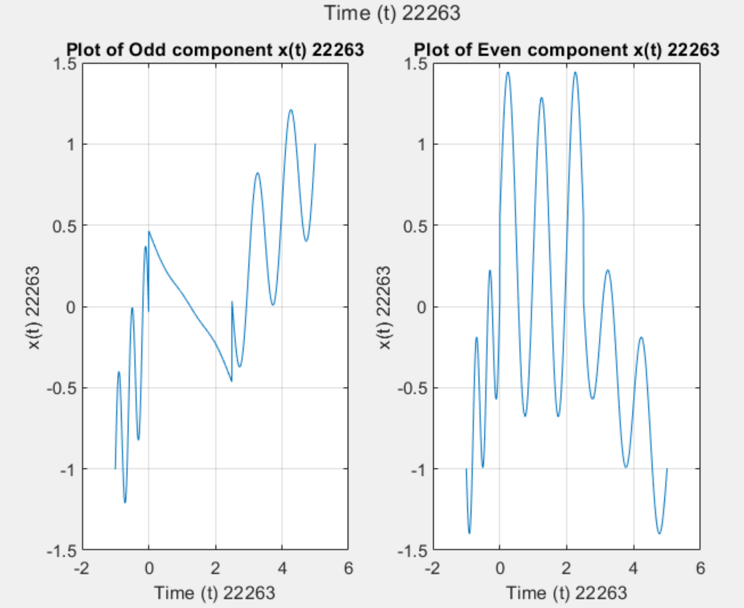



```
y1_22263 = lincomp_RollNO(1, 1, x1_22263, x2);

% Generate y2[n] = x1[n] - x2[n]
y2_22263 = lincomp_RollNO(1, -1, x1_22263, x2);

% Plot x1[n] and x2[n]
subplot(2, 2, 1);
stem(x1_22263, 'filled');
title('x1[n] 22263');
xlabel('n 22263');
ylabel('Amplitude 22263');

subplot(2, 2, 2);
stem(x2, 'filled');
title('x2[n] 22263');
xlabel('n 22263');
ylabel('Amplitude 22263');

% Plot y1[n] and y2[n]
subplot(2, 2, 3);
stem(y1_22263, 'filled');
title('y1[n] = x1[n] + x2[n] 22263');
xlabel('n 22263');
ylabel('Amplitude 22263');

subplot(2, 2, 4);
stem(y2_22263, 'filled');
title('y2[n] = x1[n] - x2[n] 22263');
xlabel('n 22263');
ylabel('Amplitude 22263');
```

Result/Observation:
We can observe that as we change the values of a and b we get the output graphs for $y1\,[n] = x1\,[n] + x2[n]$ and $y2\,[n] = x1\,[n] - x2[n]$ respectively.







7. For the signal

$$x(t) = \begin{cases} 2t & -1 < t \le 0 \\ e^{-t} + \sin(2\pi t) & 0 < t < 5 \end{cases}$$

a) Sketch x(t)

b) Plot x(0.5*t+0.5)

c) Compute the odd and even components of x(t) and plot them

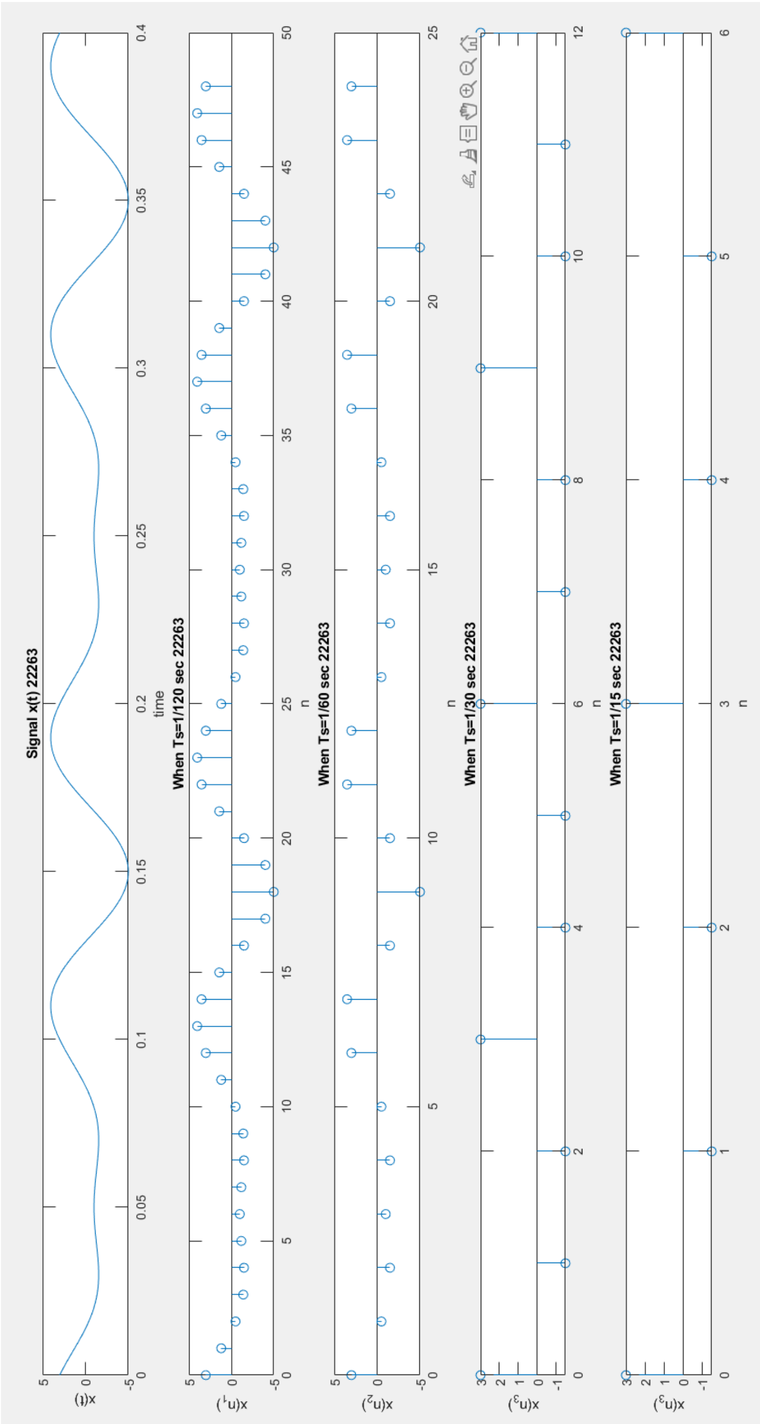
Code:

```
% Define the time vectors
t1_22263 = linspace(-1, 0, 500); % For 2t
t2_22263 = linspace(0, 5, 1000); % For e^(-t) +
sin(2*pi*t)
t_22263 = [t1_22263, t2_22263(2:end)]; % Combine
and avoid duplicating t=0

% Define x(t)
x1_22263 = 2*t1_22263;
x2_22263 = exp(-t2_22263) + sin(2*pi*t2_22263);
x_22263 = [x1_22263 x2_22263(2:end)]; % Combine

% Plot x(t)
figure;
plot(t_22263, x_22263);
xlabel('Time (t) 22263');
ylabel('x(t) 22263');
title('Plot of x(t) 22263');
grid on;

% Define the modified time vector and compute
x(0.5t + 0.5)
t_mod_22263 = 0.5*t_22263 + 0.5;
x_mod_22263 = interp1(t_22263, x_22263,
t_mod_22263, 'linear', 'extrap'); % Interpolate x
at modified time points

% Plot x(0.5t + 0.5)
figure;
plot(t_22263, x_mod_22263);
xlabel('Time (t) 22263');
ylabel('x(0.5t + 0.5) 22263');
title('Plot of x(0.5t + 0.5) 22263');
grid on;

%Plotting Even and Odd Components
x_odd_22263=(x_22263-fliplr(x_22263))/2;
x_even_22263=(x_22263+fliplr(x_22263))/2;

figure;
subplot(1,2,1);
plot(t_22263,x_odd_22263);
xlabel('Time (t) 22263');
ylabel('x(t) 22263');
```

```
title('Plot of Odd component x(t) 22263');
grid on;

subplot(1,2,2);
plot(t_22263,x_even_22263);
xlabel('Time (t) 22263');
ylabel('x(t) 22263');
title('Plot of Even component x(t) 22263');
grid on;
```

Result/Observation:

We can observe that as we change the values of a and b we get the output graphs for $y1\ [n] = x1\ [n] + x2[n]$ and $y2\ [n] = x1\ [n] - x2[n]$ respectively.



Date:5/2/2024

Work Sheet No. 2
Sampling and its effects

Aim:

To understand the effect of sampling a continuous time signal at Nyquist rate, Under sampling and Oversampling condition.

Questions:

1. Generate and plot the signal x(t) = 3cos(20πt)−2sin(30πt) over a time range of 0 < t < 400msec. Also plot the discrete time signal formed by sampling this function at the following sampling intervals:

a) Ts = 1 120 sec

b) Ts = 1 60 sec

c) Ts = 1 30 sec

d) Ts = 1 15 sec Based on your results, comment on how fast this signal should be sampled so that it could be reconstructed from the samples?

Code:

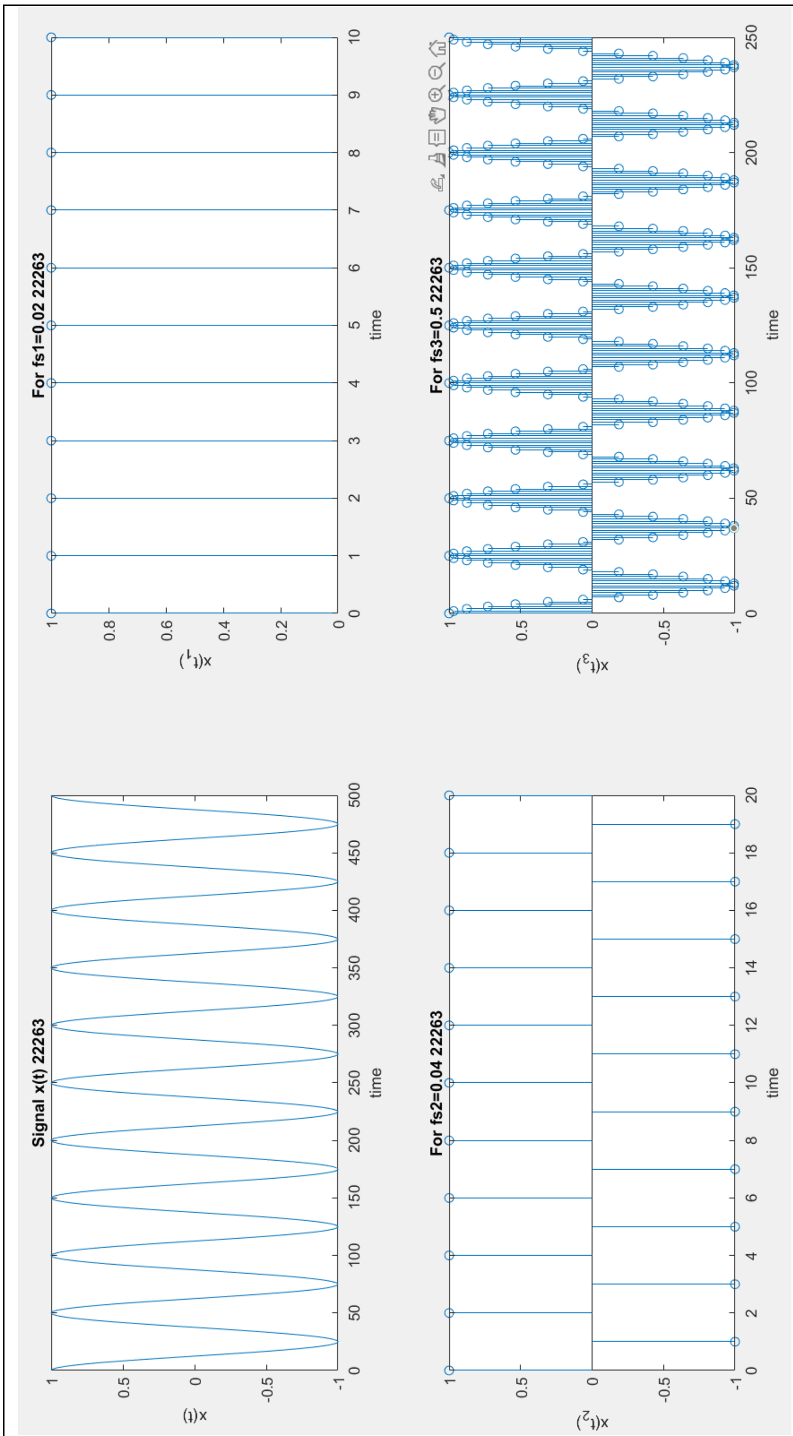
```
t_22263=0:0.001:0.4;

x_t_22263= 3*cos(20*pi*t_22263)-
2*sin(30*pi*t_22263);

subplot(5,1,1)
plot(t_22263,x_t_22263)
xlabel('time')
ylabel('x(t)')
title('Signal x(t) _22263')

t1_22263=0:(1/120):0.4;

x1_t_22263 = 3*cos(20*pi*t1_22263)-
2*sin(30*pi*t1_22263);

subplot(5,1,2)
stem(t1_22263*120,x1_t_22263)
xlabel('n')
ylabel('x(n_1)')
title('When Ts=1/120 sec _22263')

t2=0:(1/60):0.4;

x2_t = 3*cos(20*pi*t2)-2*sin(30*pi*t2);

subplot(5,1,3)
```

```
stem(t2*60,x2_t)
xlabel('n')
ylabel('x(n_2)')
title('When Ts=1/60 sec _22263')

t3_22263=0:(1/30):0.4;

x3_t_22263 = 3*cos(20*pi*t3_22263)-
2*sin(30*pi*t3_22263);

subplot(5,1,4)
stem(t3_22263*30,x3_t_22263)
xlabel('n')
ylabel('x(n_3)')
title('When Ts=1/30 sec _22263')

t4_22263=0:(1/15):0.4;

x4_t_22263 = 3*cos(20*pi*t4_22263)-
2*sin(30*pi*t4_22263);

subplot(5,1,5)
stem(t4_22263*15,x4_t_22263)
xlabel('n')
ylabel('x(n_3)')
title('When Ts=1/15 sec _22263')
```

Result/Observation:

Based on my results, this signal should be sampled greater than the Nyquist rate to prevent aliasing, Here the Nyquist rate is 30Hz, therefore fs>=30Hz in order to get a faster reconstructed sample

2.A signal $x(t) = \cos(2\pi f mt)$ with fm=0.02 is sampled at frequencies a) fs1 = 0.02 b) fs2 = 0.04 c) fs3 = 0.5 Generate the signal x(t) and the discrete time signals x[n] for the corresponding sampling frequencies.



Code:

```
time_22263=0:0.01:500;

fm_22263=0.02;
fs1=fm_22263;
fs2_22263=0.04;
fs3_22263=0.5;

xoft_22263=cos(2*pi*fm_22263*time_22263);

subplot(2,2,1)
plot(time_22263,xoft_22263);
xlabel('time');
ylabel('x(t)');
title('Signal x(t) 22263');

time1_22263=0:(1/fs1):500;

xoft1_22263=cos(2*pi*fm_22263*time1_22263);

subplot(2,2,2)
stem(time1_22263*0.02,xoft1_22263)
xlabel('time');
ylabel('x(t_1)');
title('For fs1=0.02 22263');

time2_22263=0:(1/fs2_22263):500;

xoft2_22263=cos(2*pi*fm_22263*time2_22263);

subplot(2,2,3)
stem(time2_22263*0.04,xoft2_22263)
xlabel('time');
ylabel('x(t_2)');
title('For fs2=0.04 22263');

time3_22263=0:(1/fs3_22263):500;

xoft3_22263=cos(2*pi*fm_22263*time3_22263);

subplot(2,2,4)
stem(time3_22263*0.5,xoft3_22263)
xlabel('time');
ylabel('x(t_3)');
title('For fs3=0.5 22263');
```

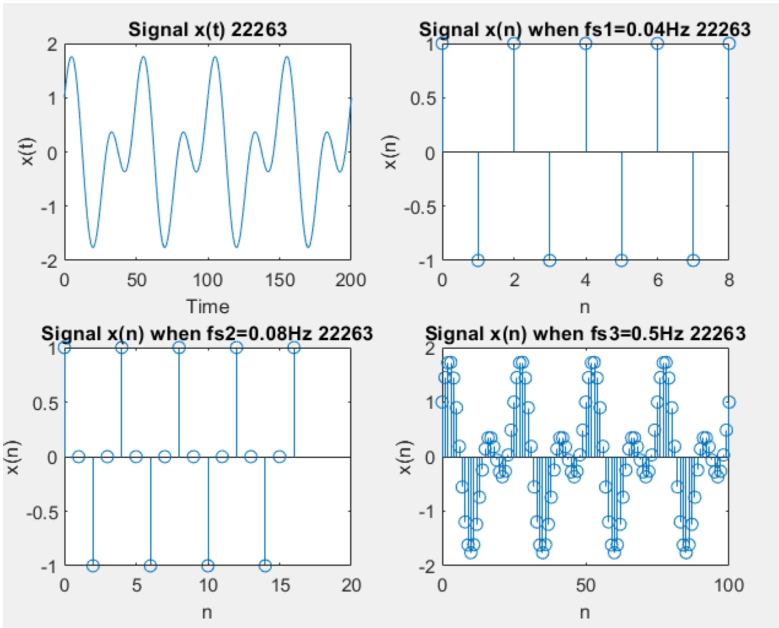
Result:

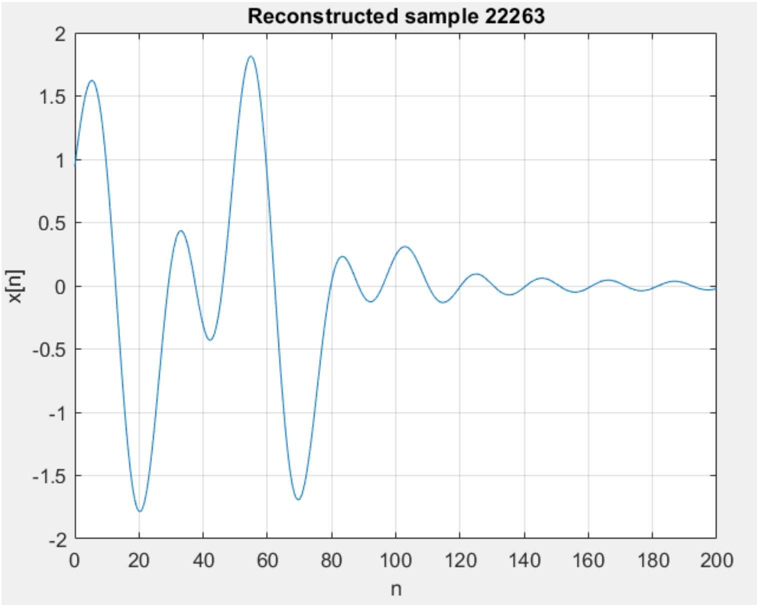
We have generated the signal x(t) and the discrete time signals x[n] for the corresponding sampling frequencies.

3.Given a signal $x(t) = \cos(0.04\pi t) + \sin(0.08\pi t)$, find the discrete time signal obtained after sampling if the sampling rate is a) fs1=0.04Hz b) fs2=0.08Hz c) fs3=0.5Hz. Generate the signal x(t) and the discrete time signals x[n] for the corresponding sampling frequencies. What do you observe?

Code:

```
Time_22263=0:0.001:200;

X_t_22263=cos(0.04*pi*Time_22263) +
sin(0.08*pi*Time_22263);

subplot(2,2,1)
plot(Time_22263,X_t_22263)
xlabel('Time')
ylabel('x(t)')
title('Signal x(t) 22263')

Fs1_22263=0.04;
Fs2_22263=0.08;
Fs3_22263=0.5;

Time1_22263=0:(1/Fs1_22263):200;

X_t1_22263=cos(0.04*pi*Time1_22263) +
sin(0.08*pi*Time1_22263);

subplot(2,2,2)
stem(Time1_22263*Fs1_22263,X_t1_22263)
xlabel('n')
ylabel('x(n)')
title('Signal x(n) when fs1=0.04Hz 22263')


Time2_22263=0:(1/Fs2_22263):200;

X_t2_22263=cos(0.04*pi*Time2_22263) +
sin(0.08*pi*Time2_22263);

subplot(2,2,3)
stem(Time2_22263*Fs2_22263,X_t2_22263)
xlabel('n')
ylabel('x(n)')
title('Signal x(n) when fs2=0.08Hz 22263')
```






```
Time3_22263=0:(1/Fs3_22263):200;

X_t3_22263=cos(0.04*pi*Time3_22263) +
sin(0.08*pi*Time3_22263);

subplot(2,2,4)
stem(Time3_22263*Fs3_22263,X_t3_22263)
xlabel('n')
ylabel('x(n)')
title('Signal x(n) when fs3=0.5Hz 22263')

sum_22263 =0;
for i=0:2:100
sum_22263 = sum_22263 + ((cos(0.04*pi*i) +
sin(0.08*pi*i)).* sinc((0.3/pi)*(Time_22263-i)));
end
X_t4_22263 = ((0.3/pi)*2)*sum_22263;
figure;
plot(Time_22263,X_t4_22263);
title('Reconstructed sample 22263');
xlabel('n');
ylabel('x[n]');
grid on;
```

<u>Result/Observation:</u>

At frequencies greater than Nyquist rate which in this case is 0.08, we can see a perfectly or more accurately reconstructed signal in the discrete time domain if we see the second graph in column 2. For fs=0.04 we see that aliasing occurs and for fs=0.08 we can see that it is not sampled very accurately but is better than the first graph. Also the final reconstructed signal is also observed.

Date:19/2/2024

Work Sheet No. 3
Sampling and its effects

Aim:

To understand the effect of sampling a continuous time signal at Nyquist rate, Under sampling and Oversampling condition.

Questions:

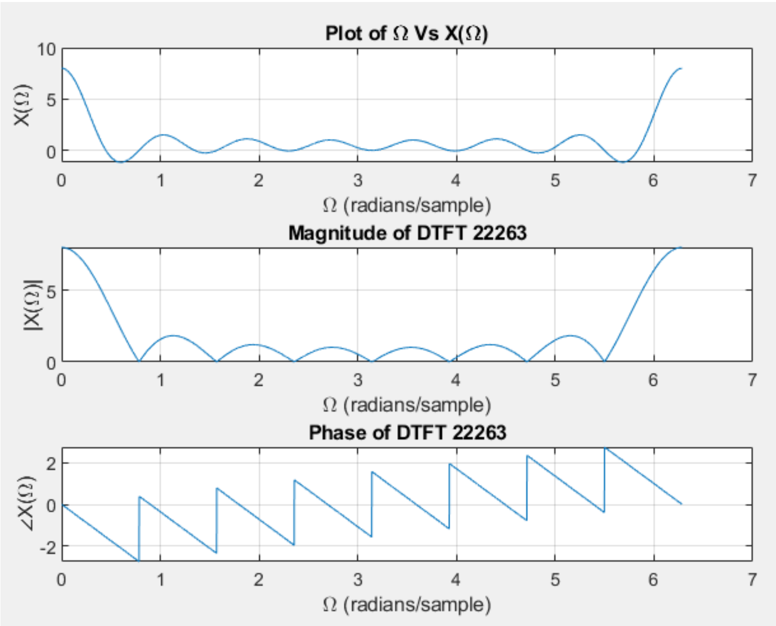
1. Write a function to calculate the DTFT of a given signal given by $X(\Omega) = \sum x[n]e\ N{-}1\ {-}j\Omega n\ n{=}0$. (Hint: Take Ω , n and x[n] as input to the function) and use it to compute the DTFT of the signal x[n]=u[n]-u[n-8].

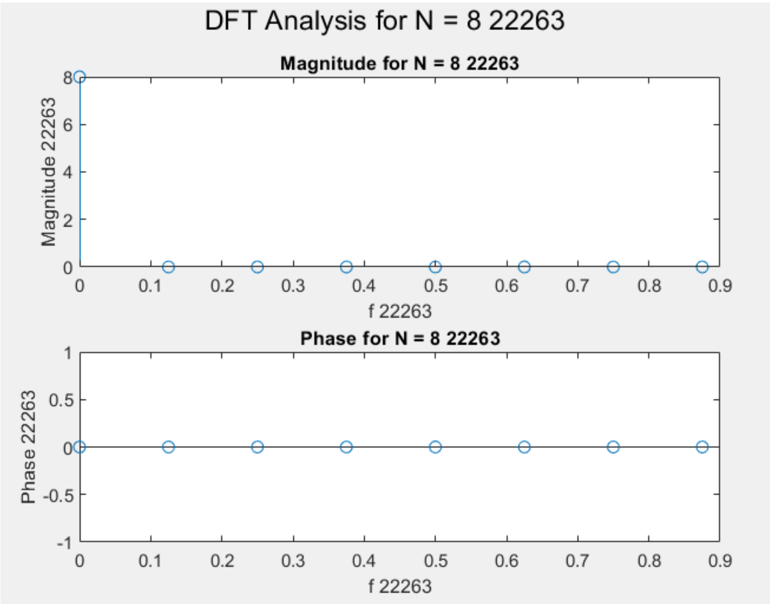
Code(Function):

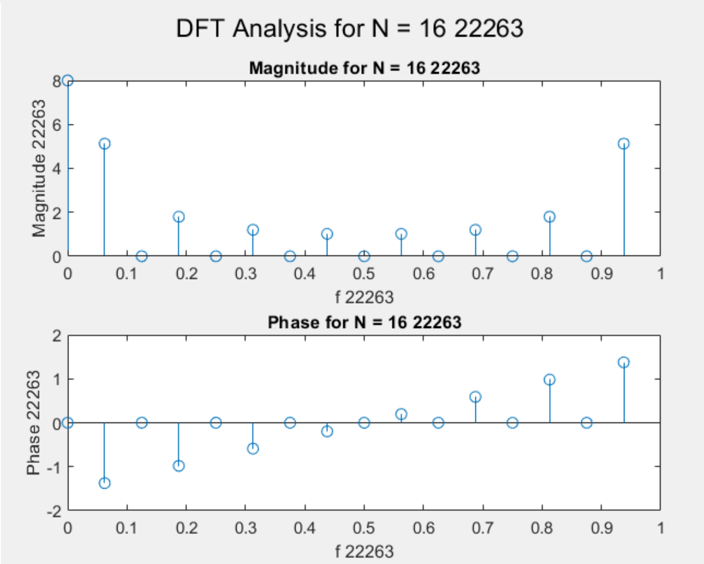
```
function X_22263=dtft(x,n,omega)
M_22263=length(omega);
N_22263= length(n);
X_22263=zeros(1,M_22263);


for m=1:M_22263
    for i=1:N_22263
            X_22263(m)=X_22263(m)+x(i)*exp(-
1j*omega(m)*n(i));
    end
end
```

Code(Main):

```
n_22263=0:10;
omega_22263=0:0.001:2*pi;
x_22263=[ones(1,8),zeros(1,3)];

X_22263=dtft(x_22263,n_22263,omega_22263);

subplot(3,1,1);
plot(omega_22263,X_22263);
title('Plot of \Omega Vs X(\Omega)');
xlabel('\Omega (radians/sample)');
ylabel('X(\Omega)');
grid on;

subplot(3,1,2);
plot(omega_22263,abs(X_22263));
title('Magnitude of DTFT 22263');
xlabel('\Omega (radians/sample)');
ylabel('|X(\Omega)|');
grid on;
```








```
subplot(3,1,3);
plot(omega_22263,angle(X_22263))
title('Phase of DTFT 22263');
xlabel('\Omega (radians/sample)');
ylabel('∠X(\Omega)');
grid on;
```

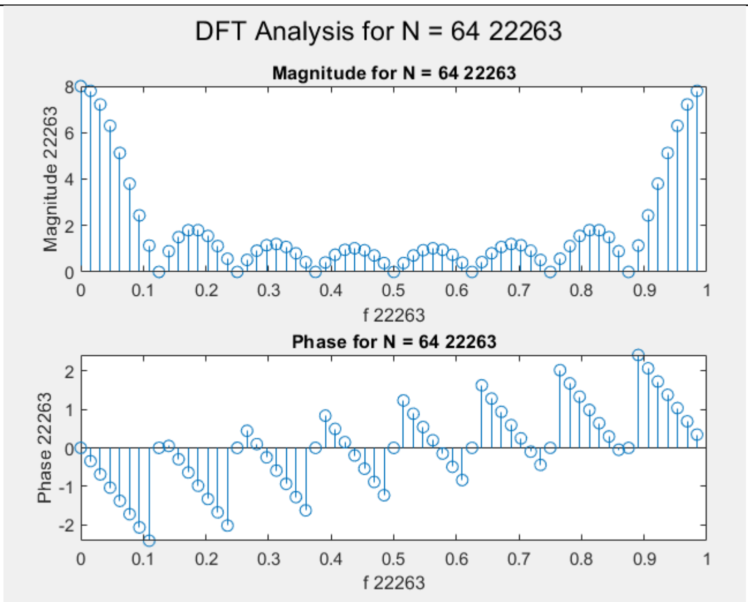
Result/Observation:

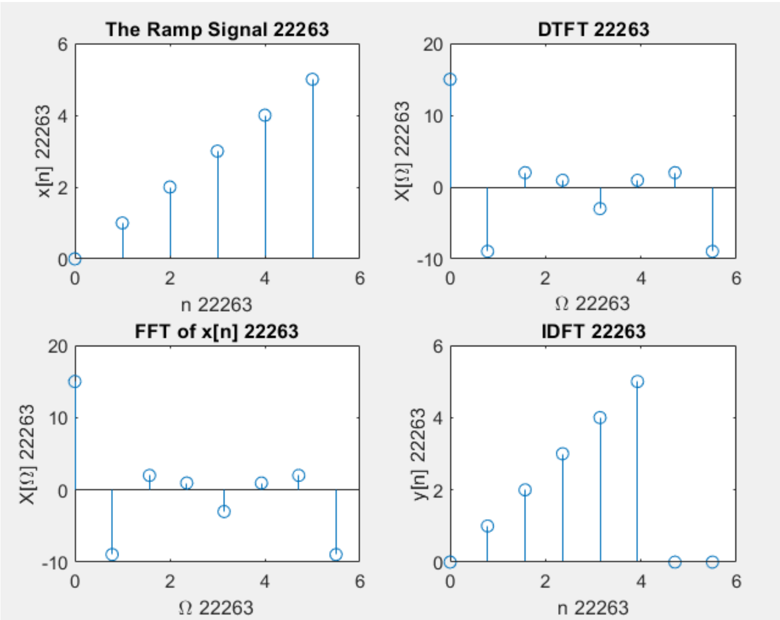
We have generated DTFT of $X(\Omega)$ and plotted the magnitude and phase spectrum.

2. A discrete time signal is given as $x[n] = 1$ for $0 \leq n \leq 7$. Find the DFT of x[n] for N=8, N=16 and N=64. (Use MATLAB command fft). Plot magnitude response and phase spectrum and observe the changes in both the spectra while changing N.

Code:

```
n_22263 = 0:7;
x_22263 = ones(size(n_22263));
N_values_22263 = [8, 16, 64];
for N = N_values_22263
X_22263 = fft(x_22263, N);
f_22263 = (0:N-1) / N;
mag_22263 = abs(X_22263);
phase_22263 = angle(X_22263);
figure;
subplot(2, 1, 1);
stem(f_22263, mag_22263);
title(['Magnitude for N = ', num2str(N),'
22263']);
xlabel('f 22263');
ylabel('Magnitude 22263');
subplot(2, 1, 2);
stem(f_22263, phase_22263);
title(['Phase for N = ', num2str(N),' 22263']);
xlabel('f 22263');
ylabel('Phase 22263');
sgtitle(['DFT Analysis for N = ', num2str(N),'
22263']);
end
```

	**<u>Result/Observation:</u>** For the three values of N we have generated the fft and we can observe the change in the magnitude and phase response for all 3 values of N. The higher the value of N the better is the magnitude and phase.



3. The sequence x[n] is a length-6 sequence defined for $0 \le n \le 5$ and given as x[n]={0,1,2,3,4,5}.

(a) Using the function prepared in Question 1, Calculate the DTFT of x[n] at eight equally spaced point given by $\Omega k = 2\pi k\ 8\ 0 \le k \le 7$. Also apply an eight-point inverse - DFT to these DFT samples to get y[n]. From the result, observe how to extract the original sequence x[n].

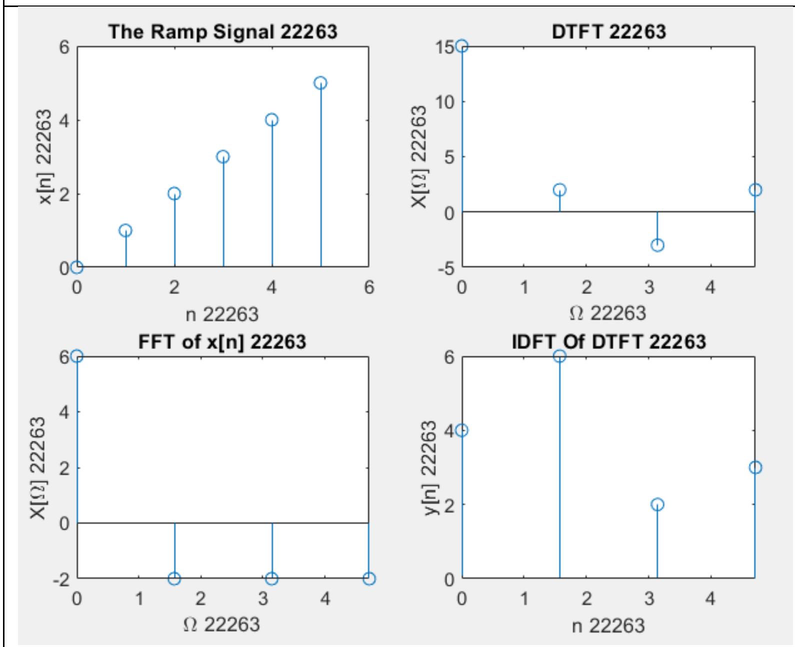
(b) Using the function prepared in Question 1, Calculate the DTFT of x[n] at four equally spaced points given by $\Omega k = 2\pi k\ 4\ 0 \le k \le 3$. Also apply a four-point inverse-DFT to these DFT samples to get y[n]. From the result, observe how to extract the original sequence x[n]. (Hint: For inverse DFT use the inbuilt function ifft)

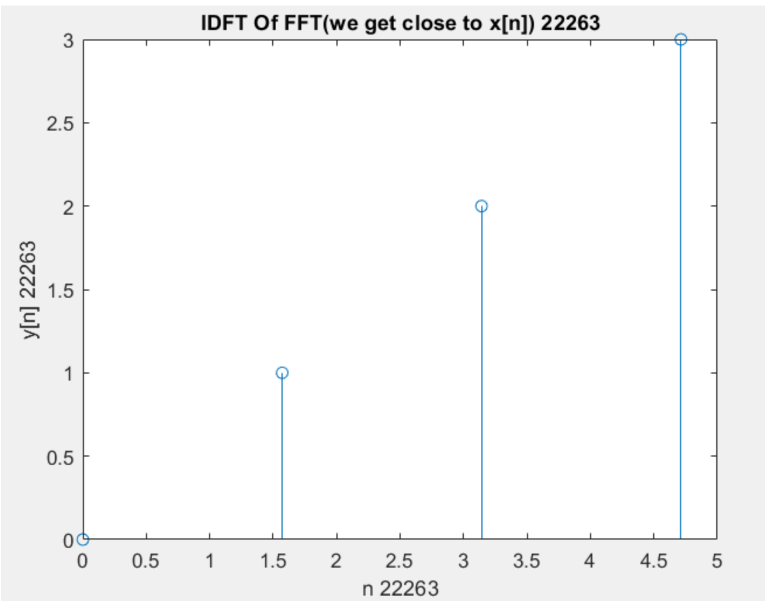
<u>Code:</u>

<u>Part a)</u>

```
n_22263=0:1:5;
x_22263=n_22263;

k_22263=0:7;
omega_22263=(2*pi*k_22263)/8;
X_22263=dtft(x_22263,n_22263,omega_22263);

X1_22263=fft(x_22263,8);
X2_22263=ifft(X_22263,8);

subplot(2,2,1);
stem(n_22263,x_22263);
xlabel('n 22263')
ylabel('x[n] 22263')
title('The Ramp Signal 22263')

subplot(2,2,2);
stem(omega_22263,X_22263);
xlabel('\Omega 22263')
ylabel('X[\Omega] 22263')
title('DTFT 22263')

subplot(2,2,3);
stem(omega_22263,X1_22263)
xlabel('\Omega 22263')
ylabel('X[\Omega] 22263')
title('FFT of x[n] 22263')
```

```
subplot(2,2,4);
stem(omega_22263,X2_22263);
xlabel('n 22263')
ylabel('y[n] 22263')
title('IDFT 22263')
```





Part b)

```
n_22263=0:1:5;
x_22263=n_22263;

k_22263=0:3;
omega_22263=(2*pi*k_22263)/4;
X_22263=dtft(x_22263,n_22263,omega_22263);

X1_22263=fft(x_22263,4);
X2_22263=ifft(X_22263,4);

subplot(2,2,1);
stem(n_22263,x_22263);
xlabel('n 22263')
ylabel('x[n] 22263')
title('The Ramp Signal 22263')

subplot(2,2,2);
stem(omega_22263,X_22263);
xlabel('\Omega 22263')
ylabel('X[\Omega] 22263')
title('DTFT 22263')

subplot(2,2,3);
stem(omega_22263,X1_22263)
xlabel('\Omega 22263')
ylabel('X[\Omega] 22263')
title('FFT of x[n] 22263')

subplot(2,2,4);
stem(omega_22263,X2_22263);
xlabel('n 22263')
ylabel('y[n] 22263')
title('IDFT 22263')
```

Result/Observation:

From the resultant graphs, we observe how to extract the original sequence x[n], which is by taking the IDFT of the DTFT in part (a), in part b if we take the IDFT of FFT we get x[n] to some extent but not a perfect signal.