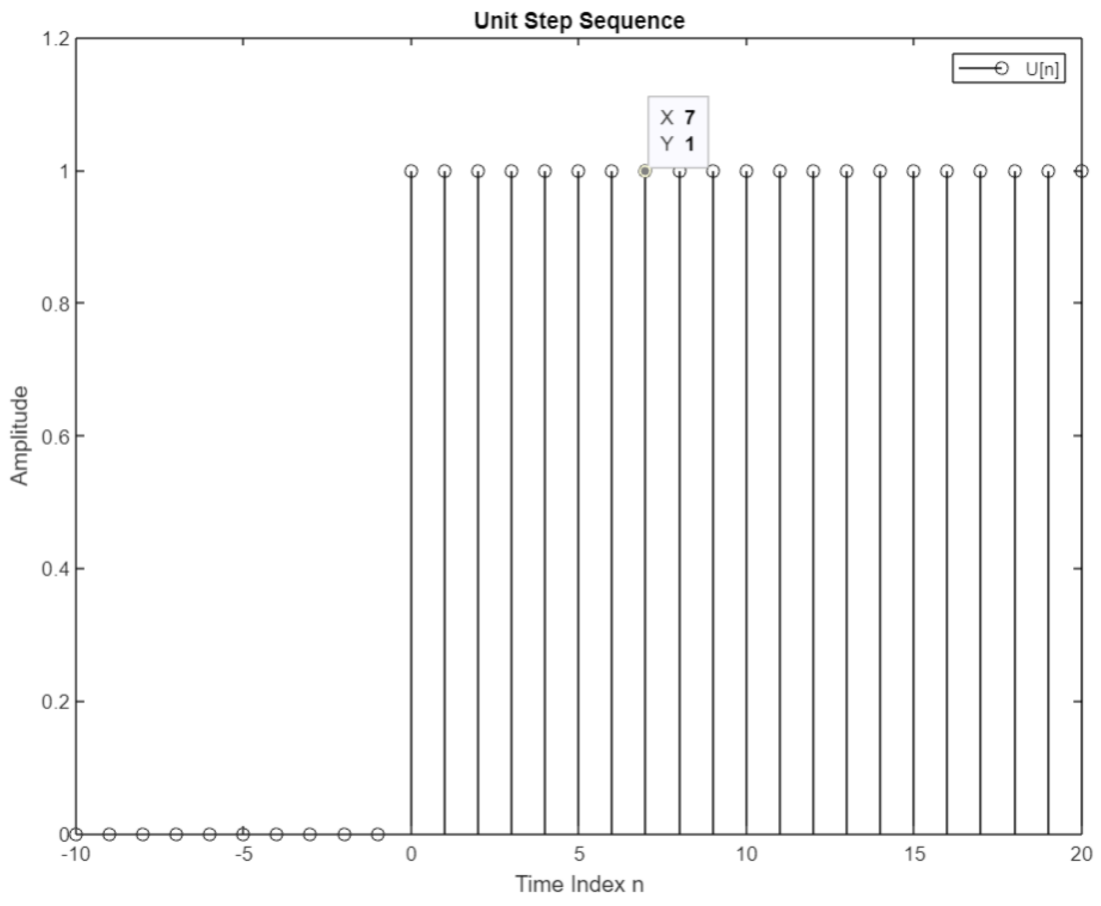
Generation of a Unit Step Sequence

INPUT

```
clc
clear all
close all
n= -10:20;
u=[zeros(1,10) ones(1,21)];
stem(n,u,'k');
xlabel('Time Index n');
ylabel('Amplitude');
title('Unit Step Sequence');
axis([-10 20 0 1.2]);
legend('U[n]');
```

OUTPUT



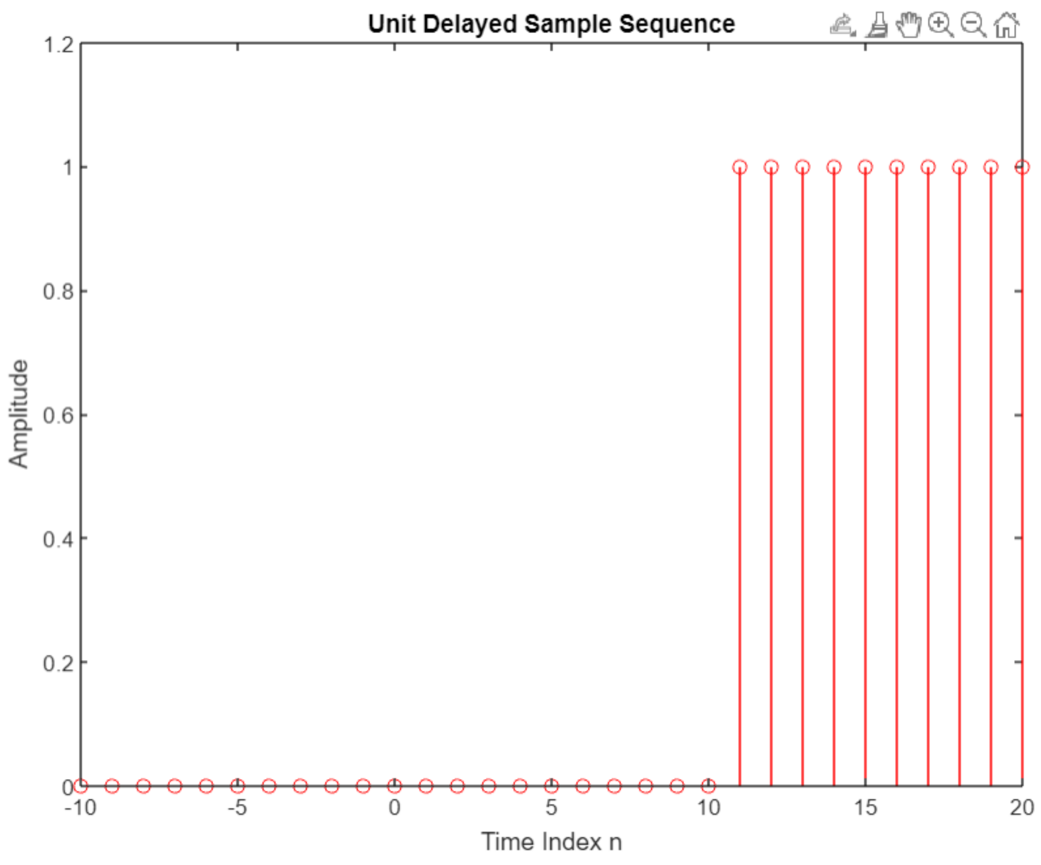
Question 2: Run A Program to Generate the Delayed Unit Step Sequence s(n) with a

- Delay of 11 sample and display it.
- Generation of a Unit Sample Sequence

INPUT

```
clc;
clear all;
close all;
%Generate a vector from -10 to 20
n=-10:20;
%Generation of a Unit Sample Sequence
u=[zeros(1,21) ones(1,10)];
%Plot The Unit Sample Sequence
stem(n,u,'r')
xlabel ('Time Index n');
ylabel('Amplitude');
title('Unit Delayed Sample Sequence')
axis([-10 20 0 1.2]);
```

OUTPUT

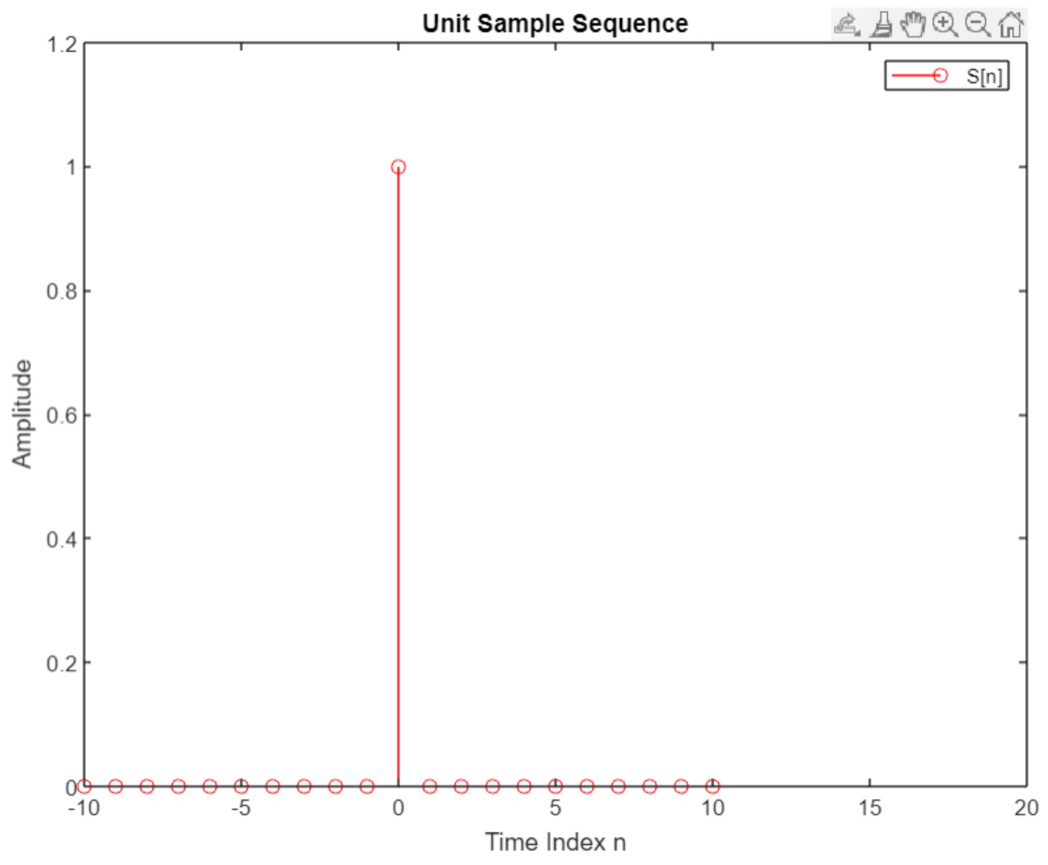


Question 3: Generation of a Unit Sample Sequence

INPUT

```
clc;
clear all;
close all;
%Generate a vector from -10 to 10
n=-10:10;
%Generation of a Unit Sample Sequence
s=[zeros(1,10) 1 zeros(1,10)];
%Plot The Unit Sample Sequence
stem(n,s,'r')
xlabel('Time Index n');
ylabel('Amplitude');
title('Unit Sample Sequence')
axis([-10 20 0 1.2]);
legend('S[n]');
```

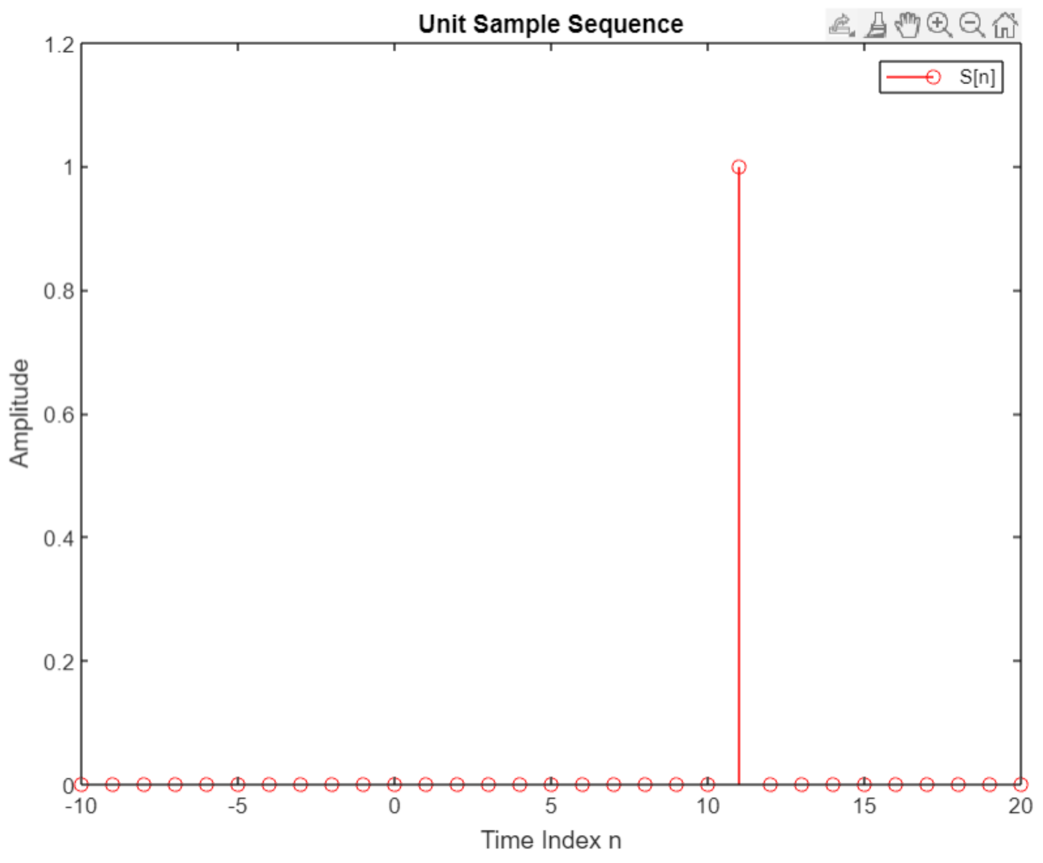
OUTPUT



INPUT

```
clc;
clear all;
close all;
%Generate a vector from -10 to 20
n=-10:20;
%Generation of a Unit Sample Sequence
s=[zeros(1,21) 1 zeros(1,9)];
%Plot The Unit Sample Sequence
stem(n,s,'r')
xlabel('Time Index n');
ylabel('Amplitude');
title('Unit Sample Sequence')
axis([-10 20 0 1.2]);
legend('S[n]');
```

OUTPUT

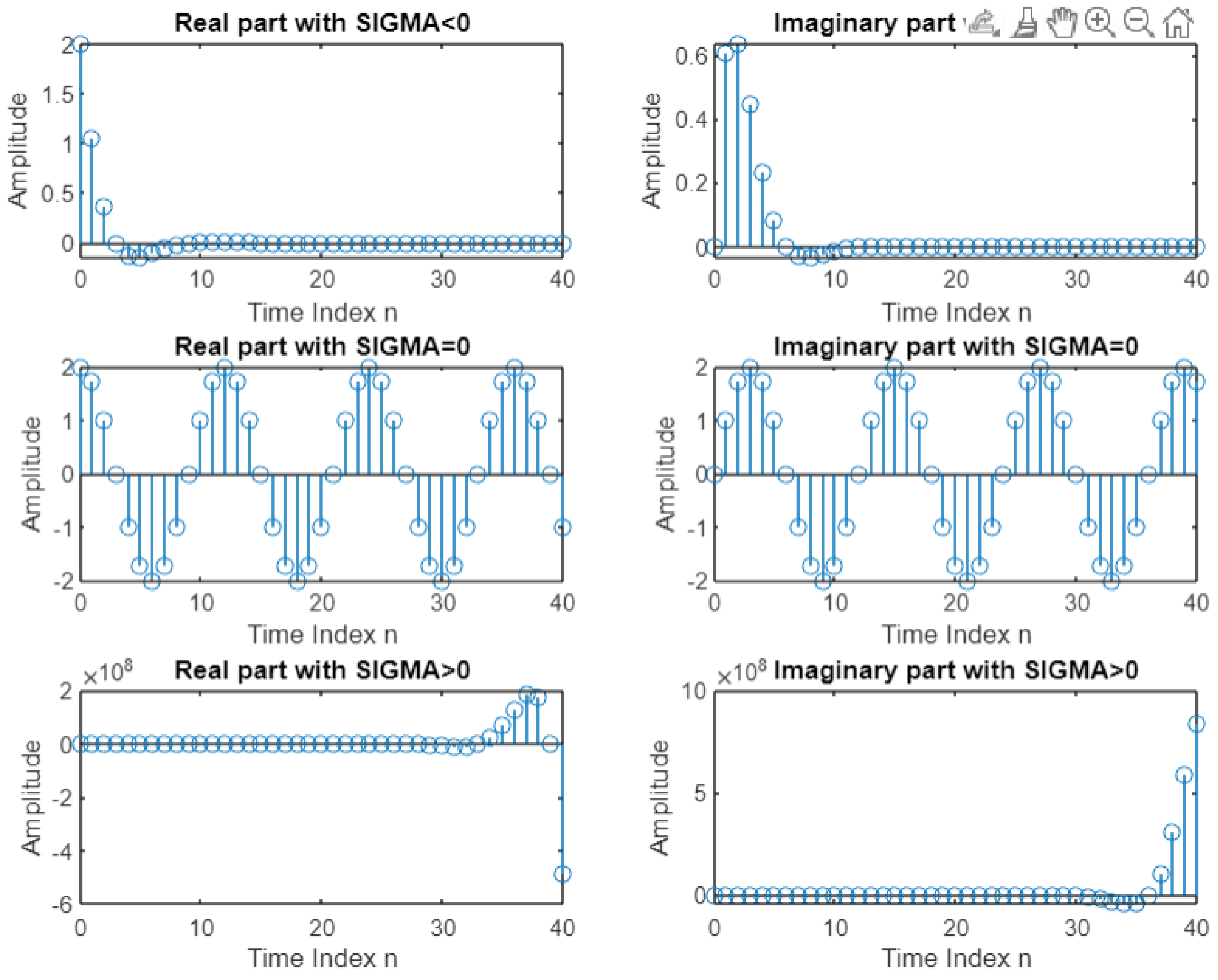


Question 4: Observe imaginary and real part of signal

INPUT

```
c=-(1/2)+(pi/6)*i;

c1=0+(pi/6)*i;

c2=(1/2)+(pi/6)*i;
k=2;
n=0:40;
x=k*exp(c*n);
x1=k*exp(c1*n);
x2=k*exp(c2*n);
subplot(3,2,1);
stem(n,real(x));
xlabel('Time Index n');
ylabel('Amplitude');
title('Real part with SIGMA<0');
subplot(3,2,2);
stem(n,imag(x));
xlabel('Time Index n');
ylabel('Amplitude');
title('Imaginary part with SIGMA<0')
subplot(3,2,3);
stem(n,real(x1));
xlabel('Time Index n');
ylabel('Amplitude');
title('Real part with SIGMA=0');
subplot(3,2,4);
stem(n,imag(x1));
xlabel('Time Index n');
ylabel('Amplitude');
title('Imaginary part with SIGMA=0')
subplot(3,2,5);
stem(n,real(x2));
xlabel('Time Index n');
ylabel ( 'Amplitude' ) ;
title('Real part with SIGMA>0');
subplot(3,2,6);
stem(n,imag(x2));
xlabel('Time Index n');
ylabel('Amplitude');
title ('Imaginary part with SIGMA>0')
```

OUTPUT

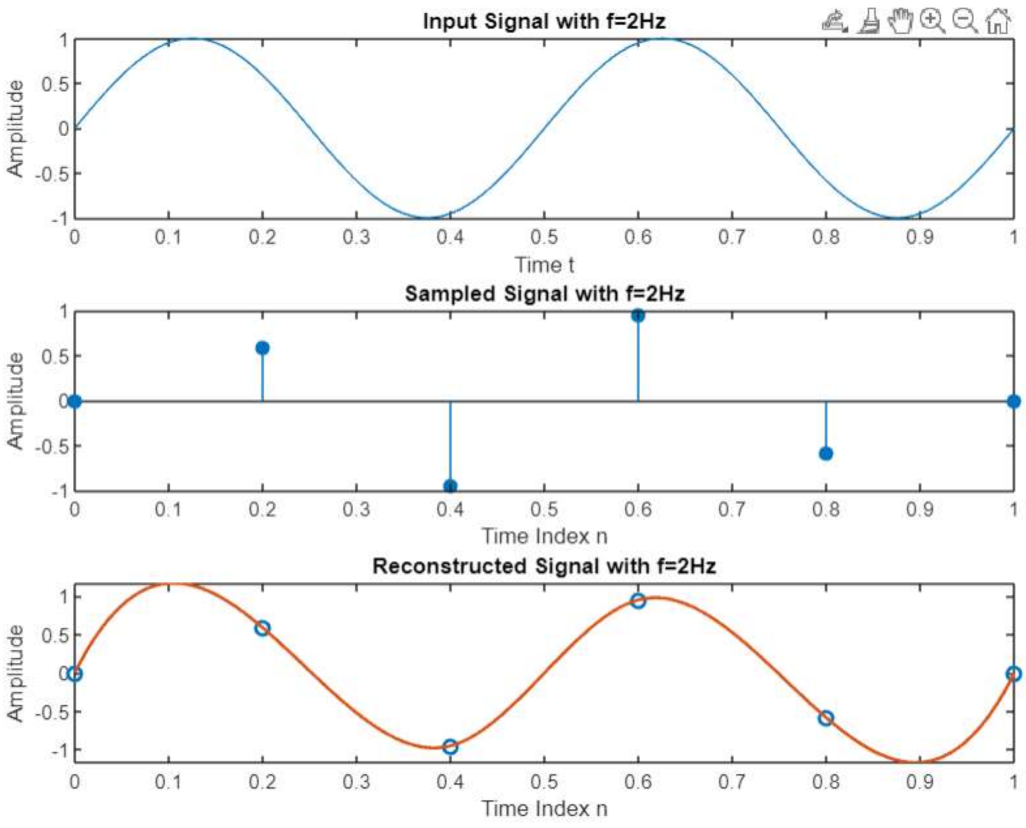

Real part with SIGMA<0
Imaginary part
Real part with SIGMA=0
Imaginary part with SIGMA=0
Real part with SIGMA>0
Imaginary part with SIGMA>0
Amplitude
Time Index n
$\times 10^8$

Question 5: Continuous time signal Generation

INPUT

```
clc;
clear all;
close all;
f=2;
T=1/f;
t=0:0.001:f*T;
y=sin(2*pi*f*t);
subplot(3,1,1);
plot(t,y);
xlabel('Time t');
ylabel('Amplitude');
title('Input Signal with f=2Hz');
%Sampling
Fs=5;
Ts=1/Fs;
n=0:Ts:f*T;
y_sampled=sin(2*pi*f*n);
subplot(3,1,2);
stem(n,y_sampled,'fill');
xlabel('Time Index n');
ylabel('Amplitude');
title('Sampled Signal with f=2Hz');
t_new=linspace(0,f*T,1001);
% t new=linspace(0,2,25);
y_reconstruct=interp1(n,y_sampled,t_new,'spline')
subplot(3,1,3);
plot(n,y_sampled,'o', t_new,y_reconstruct,'-','linewidth',1.5);
xlabel('Time Index n');
ylabel('Amplitude');
title('Reconstructed Signal with f=2Hz');
```

OUTPUT



Input Signal with f=2Hz
Amplitude
Time t
Sampled Signal with f=2Hz
Amplitude
Time Index n
Reconstructed Signal with f=2Hz
Amplitude
Time Index n

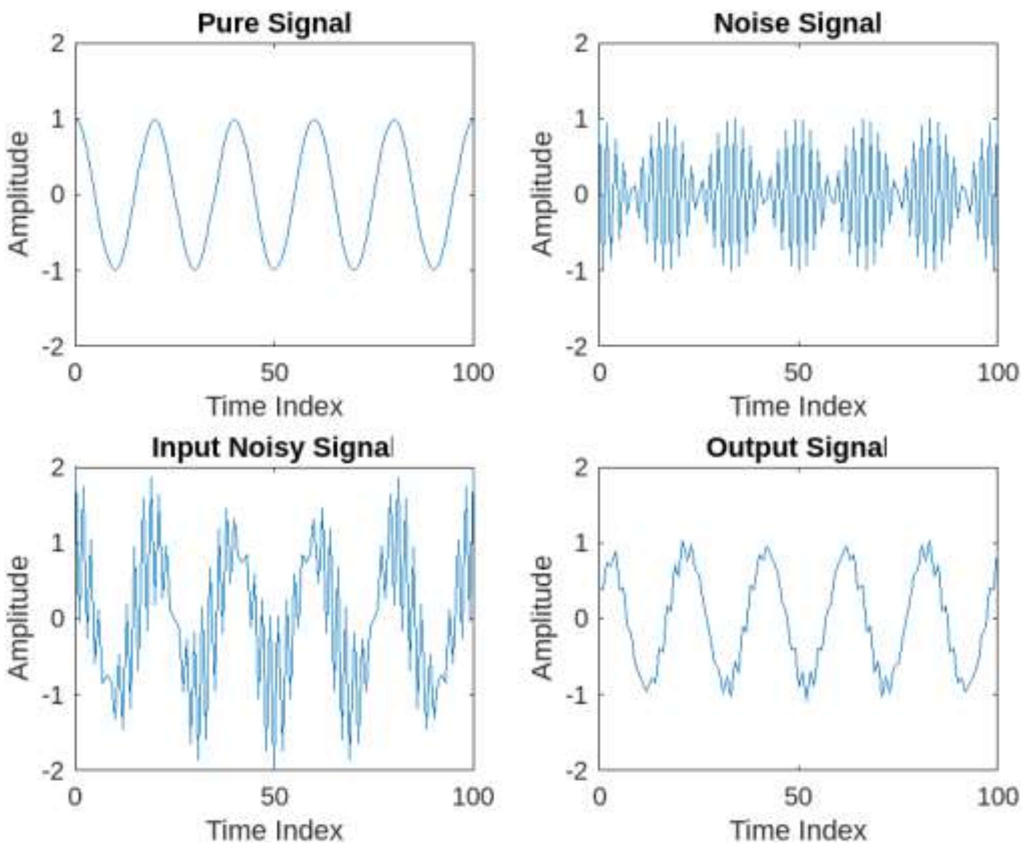
Question 6: Simulation of an M_Point Moving Average Filter

INPUT

```
clc;
clear all;
close all;
n=0:100;
s1=cos(2*pi*0.05*n); % A low frequency sinusoid
s2=cos(2*pi*0.47*n); % A high frequency sinusoid
x=s1+s2;
%Implementation of M_Point Moving Average Filter
M=input('Desired Length of The Filter = ');
num=ones(1,M);
z=filter(num,1,x);
y=filter(num,1,x)/M;
%Display the Input & Output Signals
clf;
subplot(2,2,1);
plot(n,s1);
axis([0,100 -2,2]);
xlabel('Time Index');
ylabel('Amplitude');
title ('Pure Signal');
subplot(2,2,2);
plot(n,s2);
axis([0,100 -2,2]);
xlabel ('Time Index');
ylabel('Amplitude');
title ('Noise Signal');
subplot(2,2,3);
plot(n,x);
axis([0,100 -2,2]);
xlabel ('Time Index');
ylabel('Amplitude');
title('Input Noisy Signal');
subplot(2,2,4);
plot(n,y);
axis([0,100 -2,2]);
xlabel ('Time Index');
ylabel('Amplitude');
title('Output Signal');
```

OUTPUT

Desired Length of The Filter = 5

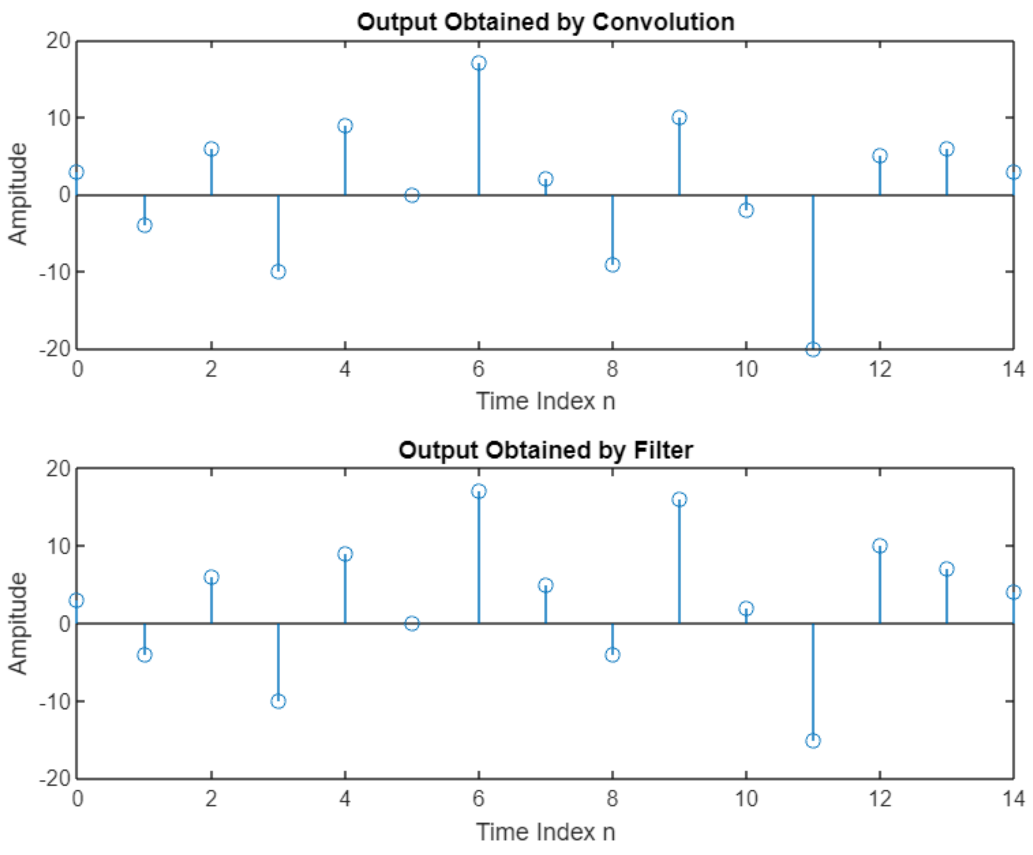


Question 7 : Convolution of FIR System

INPUT

```
clc;
clear all;
close all;
h=[3 2 1 -2 1 0 -4 0 3]; % Impulse Response
x=[1 -2 3 -4 3 2 1] ; %Input Sequence
y=conv(h,x);
n=0:14;
subplot(2,1,1);
stem(n,y);
xlabel('Time Index n');
ylabel('Ampitude');
title('Output Obtained by Convolution');
x1=[x ones(1,8)]
y1=filter(h,1,x1)
subplot(2,1,2);
stem(n,y1);
xlabel('Time Index n');
ylabel('Ampitude');
title('Output Obtained by Filter');
```

OUTPUT



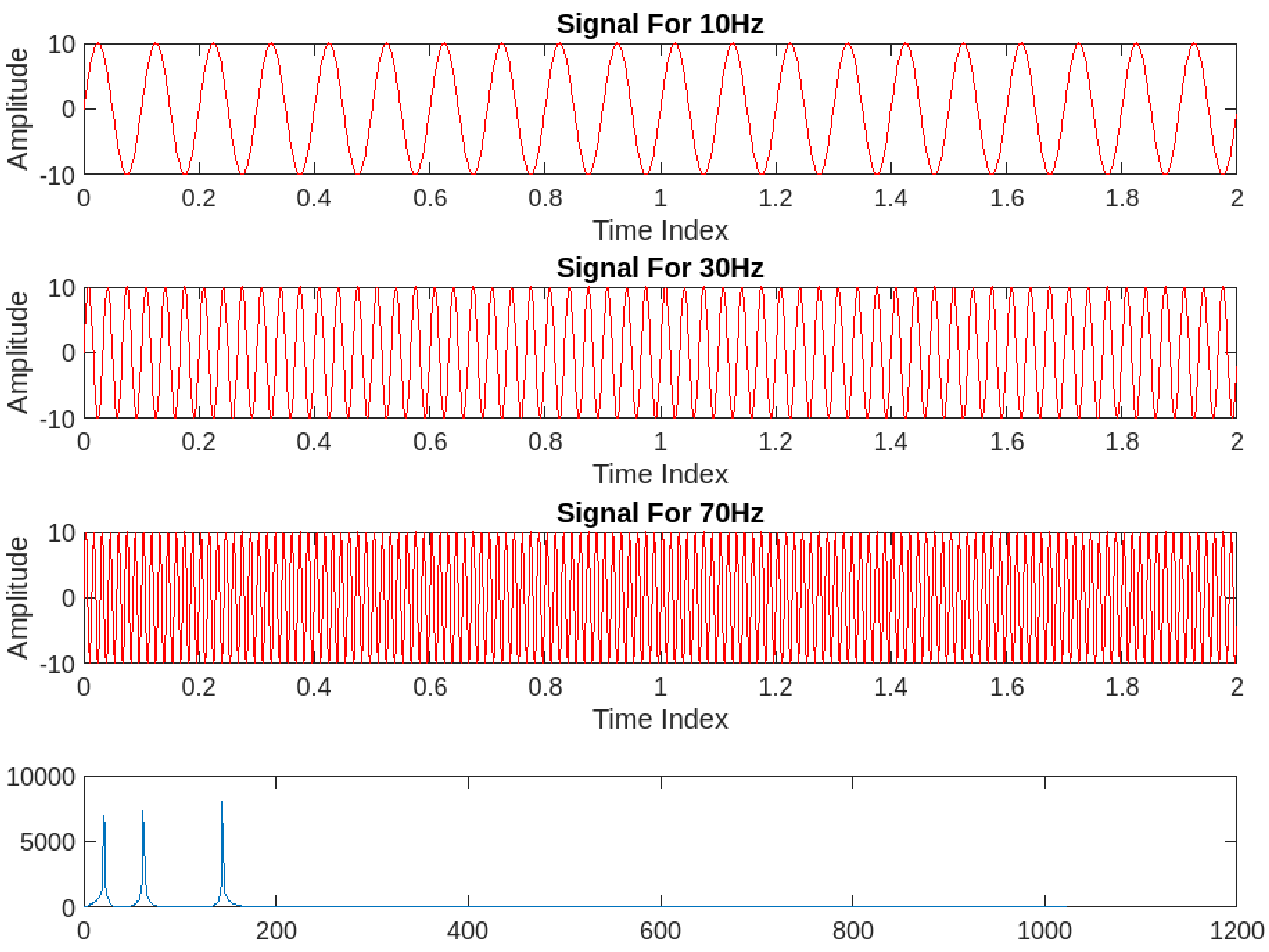
Question 8:Fast Fourier Transform Of signal

INPUT

```
clc;
clear all;
close all;
Fs=1000; %Sampling Frequency
Ts=1/Fs; %Sampling Interval
dt=0:Ts:2-Ts; %Time Duration 2 second
%Analog Signal Frequencies
f1=10;
f2=30;
f3=70;
%Analog Signals
y1=10*sin(2*pi*f1*dt);
y2=10*sin(2*pi*f2*dt);
y3=10*sin(2*pi*f3*dt);
subplot(4,1,1);
plot(dt,y1,'r');
xlabel ('Time Index');
ylabel('Amplitude');
title('Signal For 10Hz');
subplot(4,1,2);
plot(dt,y2,'r');
xlabel ('Time Index');
ylabel('Amplitude');
title('Signal For 30Hz');
subplot(4,1,3);
plot(dt,y3,'r');
xlabel('Time Index');
ylabel('Amplitude');
title('Signal For 70Hz');
%Now we combine the three signals to see thecondition of new signal
y4=y1+y2+y3;
subplot(4,1,4);
plot(dt,y4,'r');
xlabel('Time Index');
ylabel('Amplitude');
title('Signal For 10Hz,30Hz & 70Hz')
%Now we need to do FFT to recover the original frequency by drawing
%frequency Spectrum
%To do FFT we need to find out the length of the signal y4
nfft=length(y4);
%Now to get the good resolution we should to do the nextpow operation
nfft2=2^nextpow2(nfft);
ff=fft(y4,nfft2);
plot(abs(ff));

%Now to see the half of the spectrum due the symmetry we half the length
fff=ff(1:nfft2/2);
plot(abs(fff))
%Now to label the x axis to see the exact frequency we need to do the
oneline
more
```

```
xfftl=Fs* (0:nfft2/2) /nfft2;
%But in the upper xfftl there is the 1025 samples, but we need 1024
samples
xfft=Fs*(0:nfft2/2-1)/nfft2;
plot(xfft,abs(fff));
%To normalize the Amplitude
fff=10*(fff/max(fff));
plot(xfft,abs(fff));
```

OUTPUT

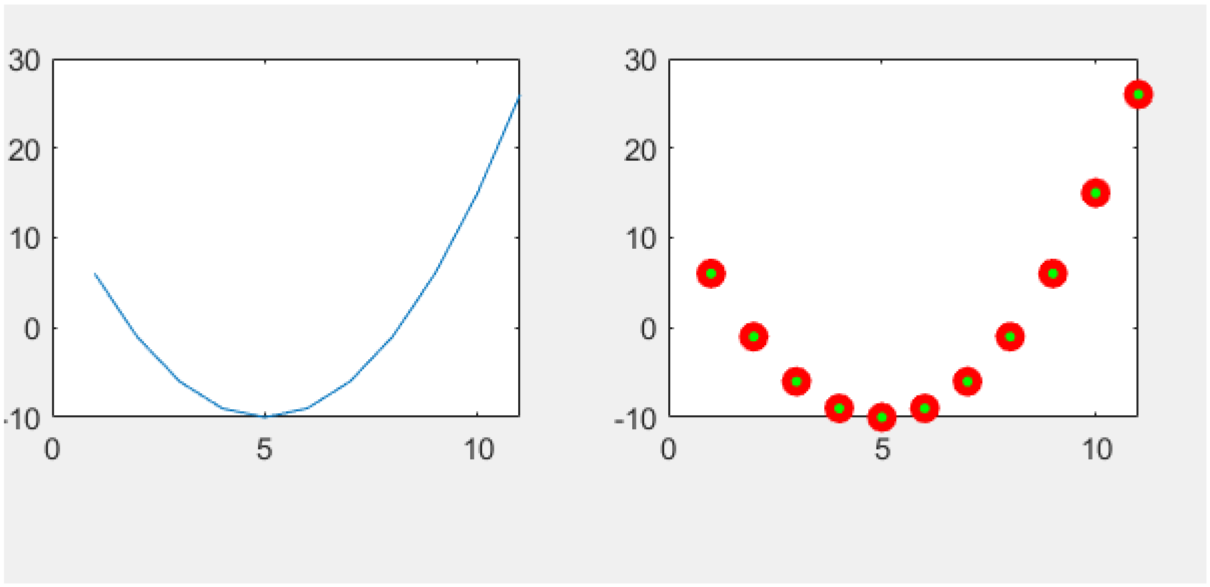


Question 9: Program to Perform Basic MATLAB Array Operations and Enhanced Plotting Techniques

INPUT

```
n1=10:-1:0;
n=0:10;
x=1:11;
y=x.*n;
y1=2^2;
y2=x.^n;
y3=ones(1,5);
y4=ones(2,5);
y5=zeros(2,5);
y6=size(n);%size shows the dimension of array or matrices.
y7=length(n);
%plot y=x^2-10x+15, where x is 1 to 11
y8=x.^2-10.*x+15;
%plot(x,y8)
%Enhanced Control of Plotted lines
%plot(x,y8,'--
ko','LineWidth',3.0,'MarkerSize',6,'MarkerEdgeColor','r','MarkerFaceCo
lor','g')
%figure(1)
%plot(x,y8)
%figure(2)
%plot(x,y8,'-
ko','LineWidth',3.0,'MarkerSize',6,'MarkerEdgeColor','r','MarkerFaceCo
lor','g')
subplot(2,2,1);
plot(x,y8);
subplot(2,2,2);
plot(x,y8,'or','LineWidth',3.0,'MarkerSize',6,'MarkerEdgeColor','r','M
arkerFaceColor','g')
y9=linspace(0,10,50);
%Divide 10 number of data into 25 divisions
```

OUTPUT

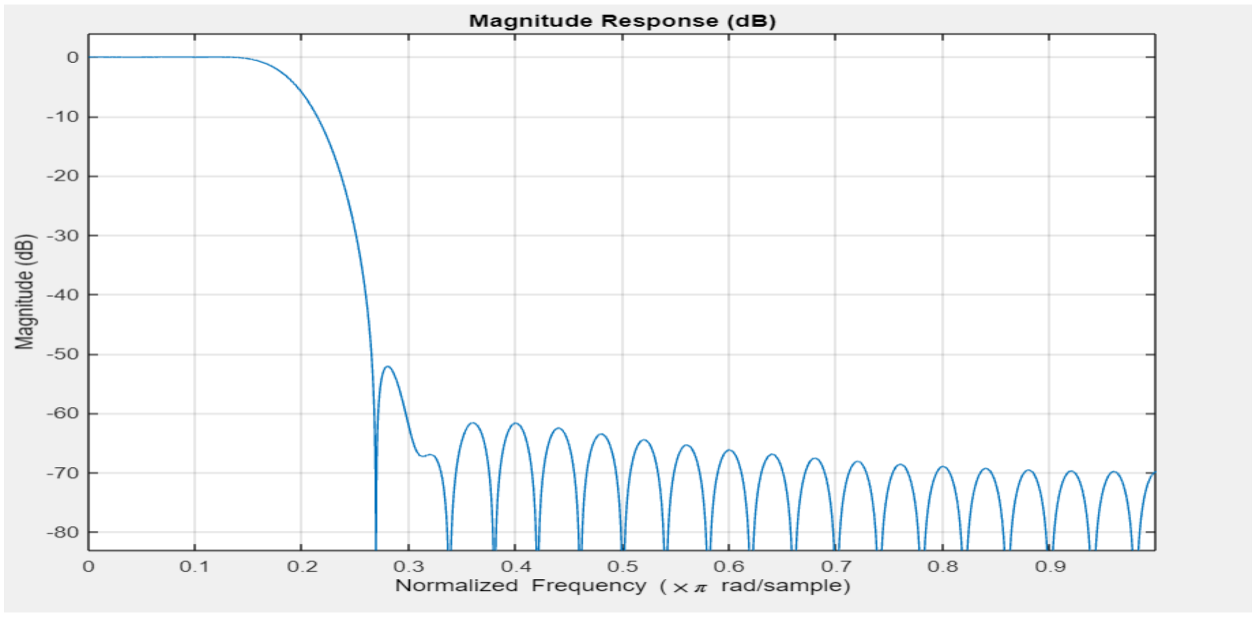


Question 10: Design and Application of Low-Pass FIR Filter Using Hamming Window in MATLAB

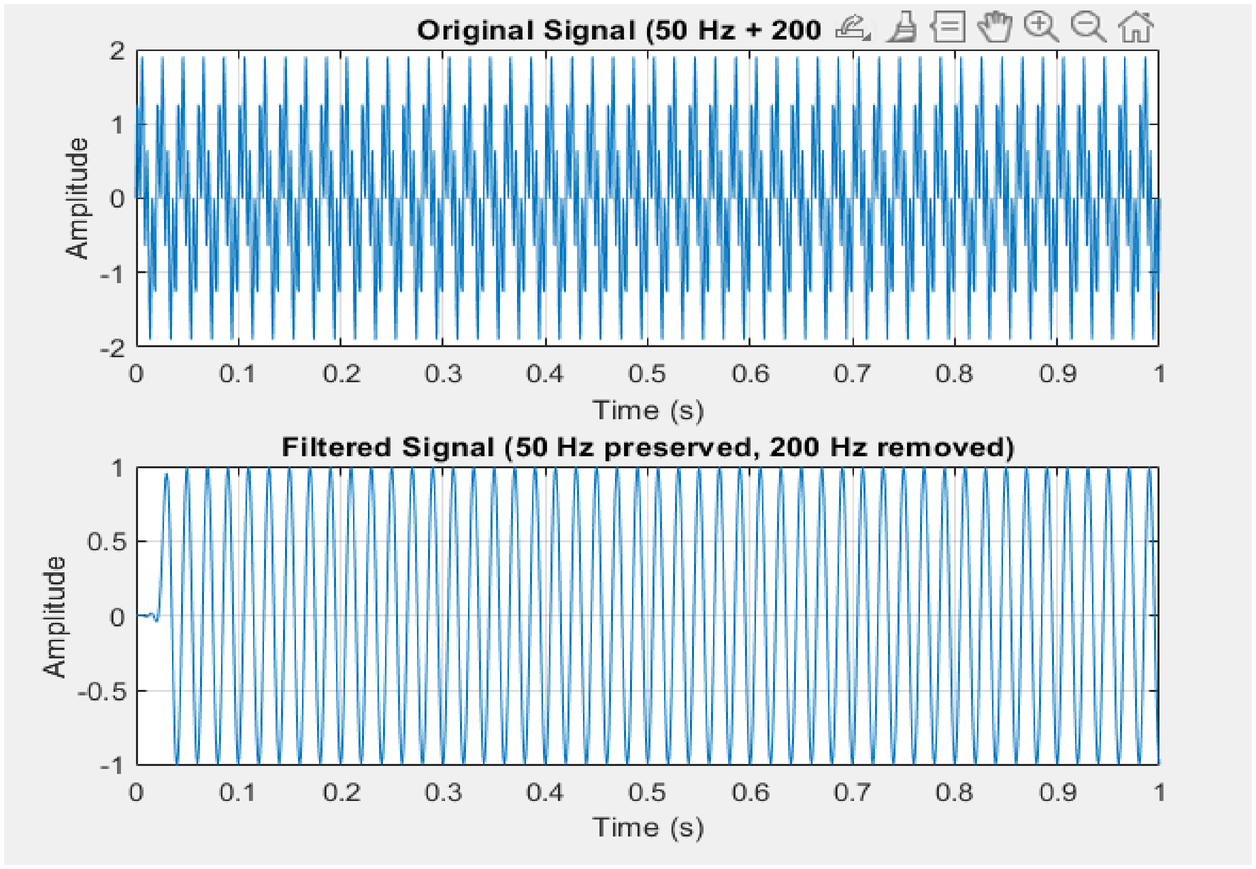
INPUT

```
clc;

clear;
close all;
% Sampling frequency
Fs = 1000; % 1000 Hz
% Cutoff frequency (Normalized)
Fc = 100; % 100 Hz cutoff
N = 50; % Filter order (Number of coefficients)
% Design Low-pass FIR filter using Hamming window,[Fc/(Fs/2)] is a
% normalize cutt off frequecncy
b = fir1(N, Fc/(Fs/2), 'low', hamming(N+1));
% Frequency response of the filter
fvtool(b,1); % Visualize filter response
% Generate a test signal (sum of two sinusoids)
t = 0:1/Fs:1; % 1 second time vector
x = sin(2*pi*50*t) + sin(2*pi*200*t); % 50Hz and 200Hz signals
% Apply FIR filter
y = filter(b,1,x);
% Plot results
subplot(2,1,1);
plot(t, x);
title('Original Signal (50 Hz + 200 Hz)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(2,1,2);
plot(t, y);
title('Filtered Signal (50 Hz preserved, 200 Hz removed)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

OUTPUT



Magnitude Response (dB)
Magnitude (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Normalized Frequency (×π rad/sample)



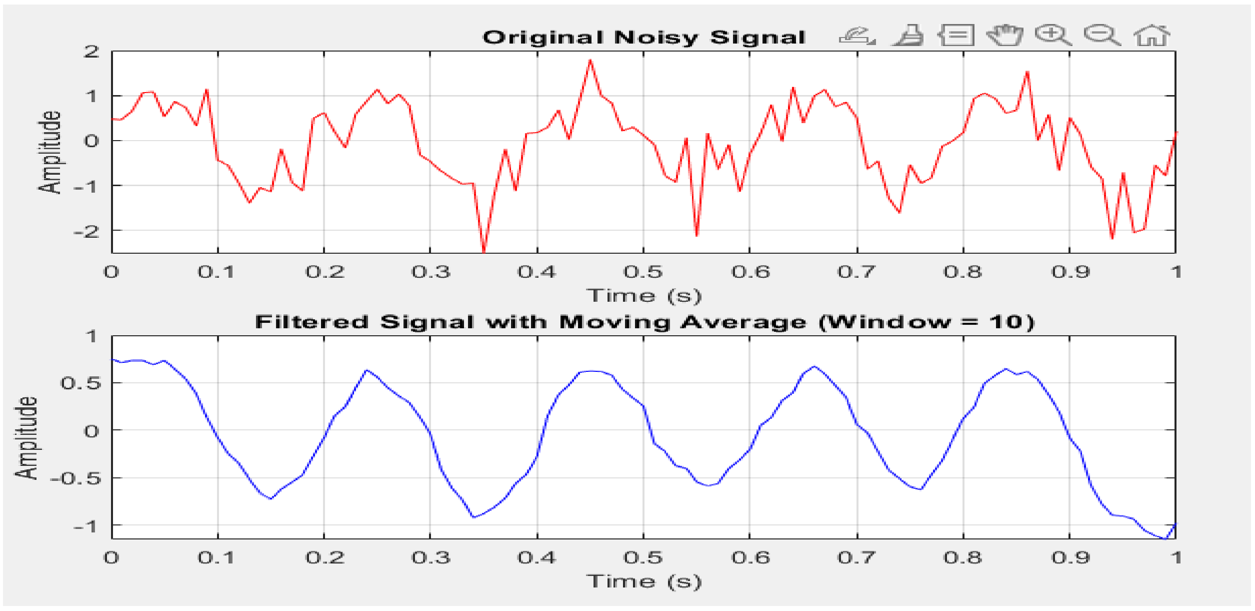
Original Signal (50 Hz + 200
Amplitude
Time (s)
Filtered Signal (50 Hz preserved, 200 Hz removed)
Amplitude
Time (s)

Question 11: Noise Reduction Using Moving Average Filter in MATLAB’

INPUT

```
clc; clear; close all;
% Generate a Noisy Signal
fs = 100; % Sampling frequency (Hz)
t = 0:1/fs:1; % Time vector (1 second duration)
signal = sin(2*pi*5*t) + 0.5*randn(size(t)); % 5 Hz sine wave with noise
% Define Moving Average Window Size
window_size = 10; % Number of samples for averaging
% Apply Moving Average Filter Using movmean()
filtered_signal = movmean(signal, window_size);
% Plot Original and Filtered Signal
figure;
subplot(2,1,1);
plot(t, signal, 'r'); hold on;
title('Original Noisy Signal');
xlabel('Time (s)'); ylabel('Amplitude');
grid on;
subplot(2,1,2);
plot(t, filtered_signal, 'b'); hold on;
title(['Filtered Signal with Moving Average (Window = ', num2str(window_size), ')']);
xlabel('Time (s)'); ylabel('Amplitude');
grid on;
```

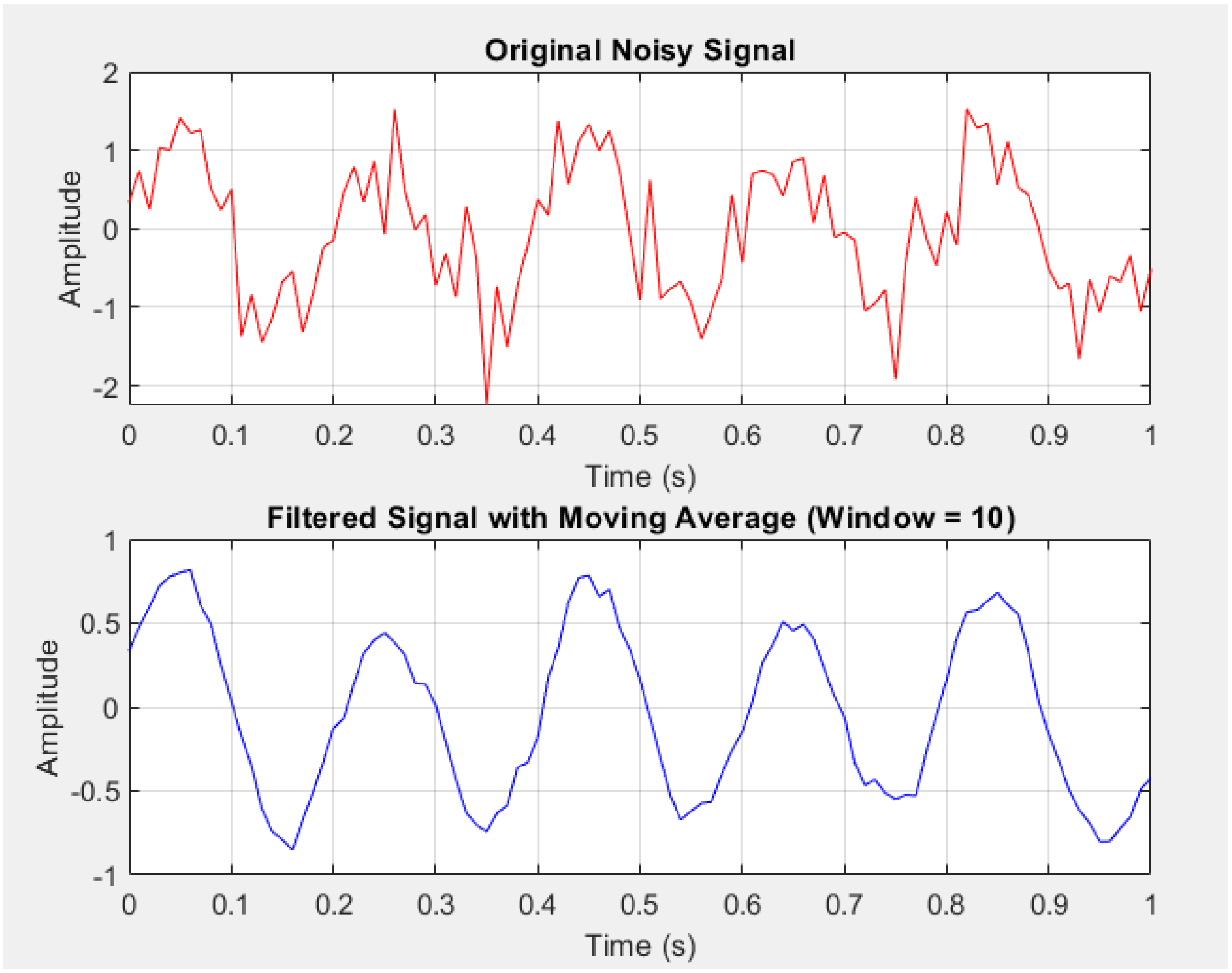
OUTPUT



Question 12: Noise Reduction Using Zero-Padded Moving Average Filter in MATLAB Input

```matlab
clc; clear; close all;
% Generate a Noisy Signal
fs = 100; % Sampling frequency (Hz)
t = 0:1/fs:1; % Time vector (1 second duration)
signal = sin(2*pi*5*t) + 0.5*randn(size(t)); % 5 Hz sine wave with noise
% Define Moving Average Window Size
window_size = 10; % Number of samples for averaging
% Zero-Pad the Signal at Both Ends
padded_signal = [zeros(1, floor(window_size/2)), signal, zeros(1,
floor(window_size/2))];
% Apply Moving Average Filter Using movmean()
filtered_signal_padded = movmean(padded_signal, window_size);
% Remove Extra Padded Values to Retain Original Length
filtered_signal = filtered_signal_padded(floor(window_size/2) + 1:end -
floor(window_size/2));
% Plot the Original and Filtered Signal
figure;
subplot(2,1,1);
plot(t, signal, 'r'); hold on;
title('Original Noisy Signal');
xlabel('Time (s)'); ylabel('Amplitude');
grid on;
subplot(2,1,2);
plot(t, filtered_signal, 'b'); hold on;
title(['Filtered Signal with Moving Average (Window = ', num2str(window_size), ')']);
xlabel('Time (s)'); ylabel('Amplitude');
grid on;
```

OUTPUT

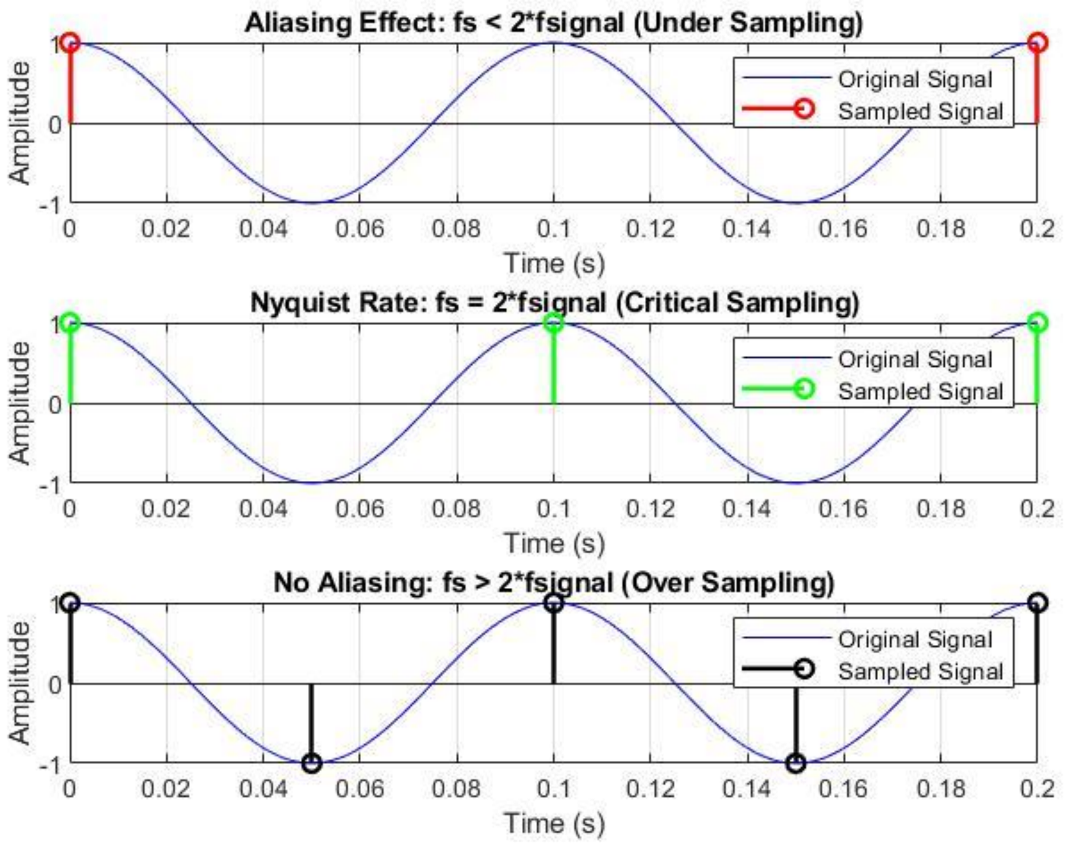

Original Noisy Signal
Amplitude
Time (s)
Filtered Signal with Moving Average (Window = 10)
Amplitude
Time (s)

Question 13: Sampling and Aliasing in Continuous-Time Signals.

Input

```
clc; clear; close all;
% Define Continuous-Time Signal (Original)
fsignal = 10;  % Original signal frequency (Hz)
t = 0:0.001:0.2; % Continuous-time axis
x = cos(2*pi*fsignal*t); % Continuous signal

% Sampling at Different Rates
fs1 = 5;   % Sampling frequency < Nyquist (Aliasing occurs)
fs2 = 20;  % Sampling frequency > Nyquist (Correct Sampling)
fs3 = 10;  % Sampling at Nyquist Rate (Borderline Case)

t1 = 0:1/fs1:0.2; % Discrete-time samples (Under Sampling)
t2 = 0:1/fs2:0.2; % Discrete-time samples (Over Sampling)
t3 = 0:1/fs3:0.2; % Discrete-time samples (Nyquist Rate)

x1 = cos(2*pi*fsignal*t1);
x2 = cos(2*pi*fsignal*t2);
x3 = cos(2*pi*fsignal*t3);

% Plot Results
figure;
subplot(3,1,1);
plot(t, x, 'b'); hold on;
stem(t1, x1, 'r', 'LineWidth', 1.5);
title('Aliasing Effect: fs < 2*fsignal (Under Sampling)');
xlabel('Time (s)'); ylabel('Amplitude');
legend('Original Signal', 'Sampled Signal');
grid on;

subplot(3,1,2);
plot(t, x, 'b'); hold on;
stem(t3, x3, 'g', 'LineWidth', 1.5);
title('Nyquist Rate: fs = 2*fsignal (Critical Sampling)');
xlabel('Time (s)'); ylabel('Amplitude');
legend('Original Signal', 'Sampled Signal');
grid on;

subplot(3,1,3);
plot(t, x, 'b'); hold on;
stem(t2, x2, 'k', 'LineWidth', 1.5);
title('No Aliasing: fs > 2*fsignal (Over Sampling)');
xlabel('Time (s)'); ylabel('Amplitude');
legend('Original Signal', 'Sampled Signal');
grid
```

Output

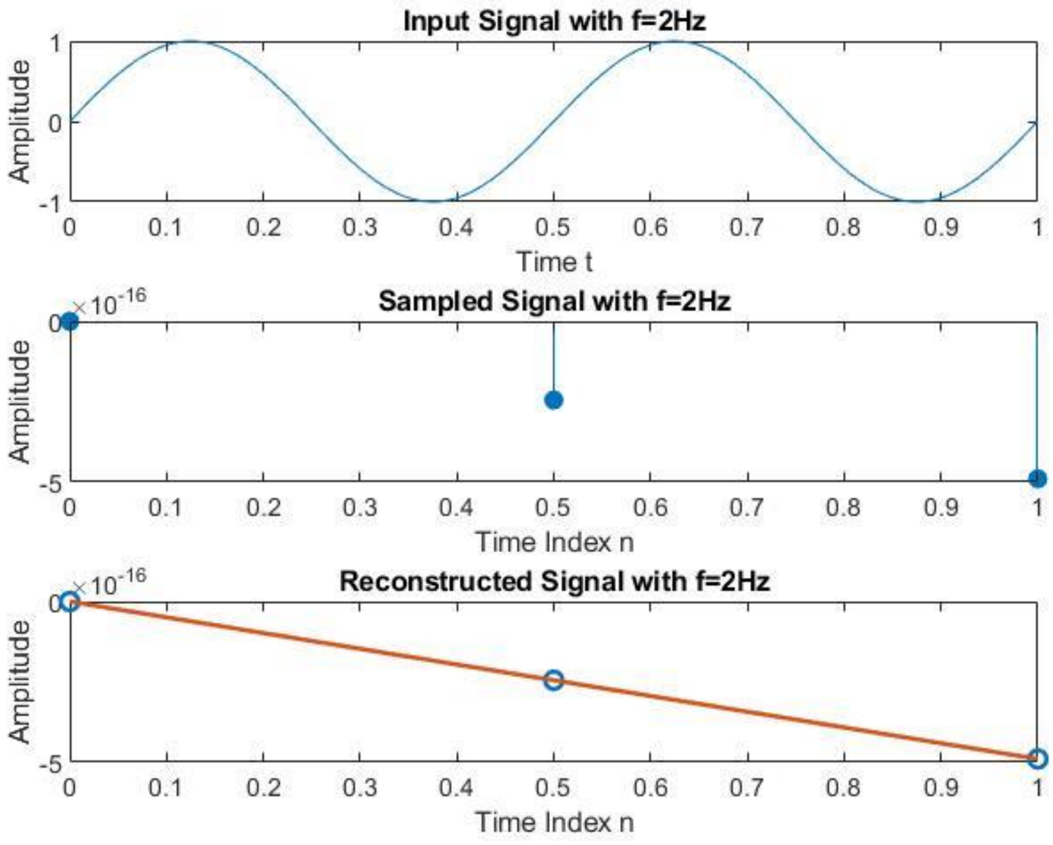

Aliasing Effect: fs < 2*fsignal (Under Sampling)
Nyquist Rate: fs = 2*fsignal (Critical Sampling)
No Aliasing: fs > 2*fsignal (Over Sampling)
Original Signal
Sampled Signal
Amplitude
Time (s)
1
0
-1
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

Question 14: Sampling and Reconstruction of a Continuous-Time Signal using Spline Interpolation

Input

```
%CT signal Generation
f=2;
T=1/f;
t=0:0.001:f*T;
y=sin(2*pi*f*t);
subplot(3,1,1);
plot(t,y);
xlabel('Time t');
ylabel('Amplitude');
title('Input Signal with f=2Hz');
%Sampling
Fs=2;
Ts=1/Fs;
n=0:Ts:f*T;
y_sampled=sin(2*pi*f*n);
subplot(3,1,2);
stem(n,y_sampled,'fill');
xlabel('Time Index n');
ylabel('Amplitude');
title('Sampled Signal with f=2Hz');
t_new=linspace(0,f*T,1001);
% t_new=linspace(0,2,25);
y_reconstruct=interp1(n,y_sampled,t_new,'spline');
subplot(3,1,3);
plot(n,y_sampled,'o', t_new,y_reconstruct,'-','linewidth',1.5);
xlabel('Time Index n');
ylabel('Amplitude');
title('Reconstructed Signal with f=2Hz');
```

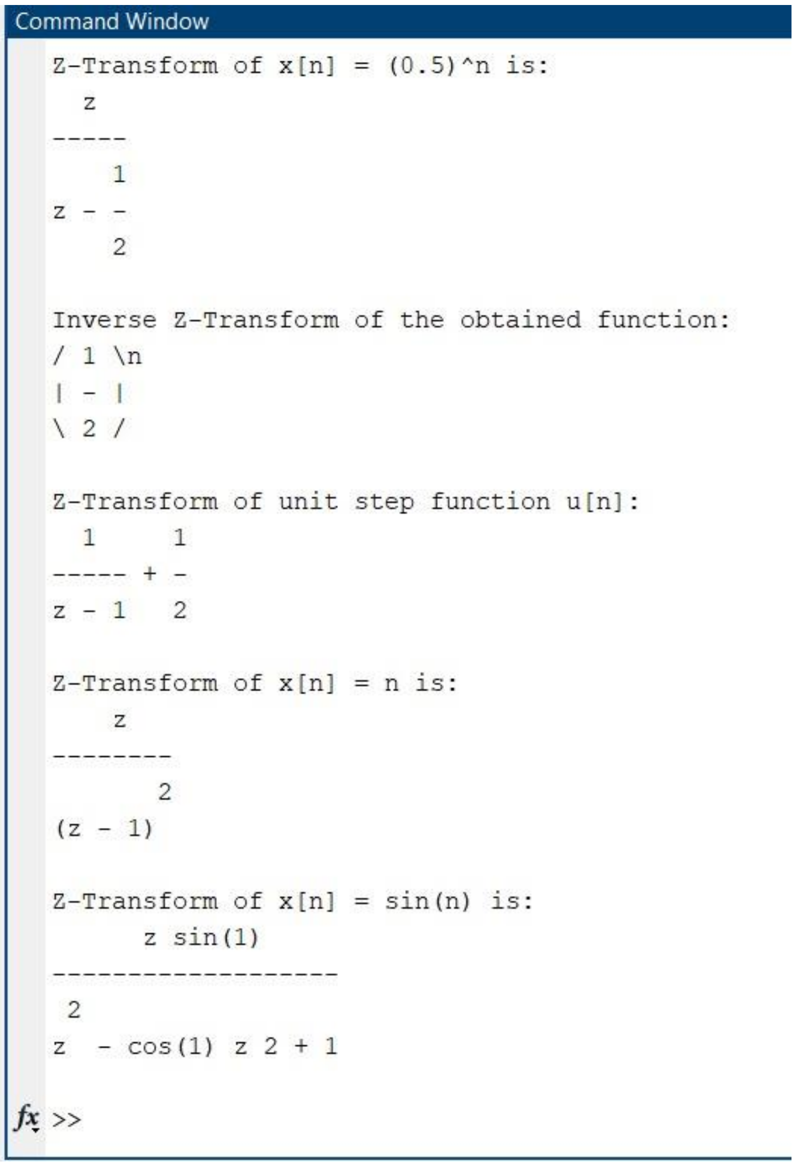
Output


Input Signal with f=2Hz
Amplitude
1
0
-1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time t
Sampled Signal with f=2Hz
×10^-16
0
-5
Amplitude
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time Index n
Reconstructed Signal with f=2Hz
×10^-16
0
-5
Amplitude
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time Index n

Question 15: Computation of Z-Transform and Inverse Z-Transform for Discrete-Time Signals

Input

```
clc;
clear;
syms n z

%% Example 1: Compute Z-Transform of a given sequence
x = (0.5)^n;  % Define the discrete-time sequence
X_z = ztrans(x, n, z); % Compute Z-Transform
disp('Z-Transform of x[n] = (0.5)^n is:');
pretty(X_z)

%% Example 2: Compute the inverse Z-Transform
X_z_inverse = iztrans(X_z, z, n); % Compute Inverse Z-Transform
disp('Inverse Z-Transform of the obtained function:');
pretty(X_z_inverse)

%% Example 3: Compute Z-Transform of a unit step function u[n]
u = heaviside(n); % Unit step function u[n]
U_z = ztrans(u, n, z);
disp('Z-Transform of unit step function u[n]:');
pretty(U_z)

%% Example 4: Compute Z-Transform of a given function x[n] = n
x2 = n; % Define sequence x[n] = n
X2_z = ztrans(x2, n, z);
disp('Z-Transform of x[n] = n is:');
pretty(X2_z)

%% Example 5: Compute Z-Transform of a Sinusoidal Signal
x3 = sin(n);
X3_z = ztrans(x3, n, z);
disp('Z-Transform of x[n] = sin(n) is:');
pretty(X3_z)
```


Command Window

```
Z-Transform of x[n] = (0.5)^n is:
  z
-----
    1
z - -
    2

Inverse Z-Transform of the obtained function:
/ 1 \n
| - |
\ 2 /

Z-Transform of unit step function u[n]:
  1     1
----- + -
z - 1   2

Z-Transform of x[n] = n is:
    z
--------
       2
(z - 1)

Z-Transform of x[n] = sin(n) is:
      z sin(1)
-------------------
 2
z  - cos(1) z 2 + 1

>>
```