PROGRAM TITLE: **1a. Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum (Using inbuilt function).**

MATLAB CODE

```
clc;
clear all;

xn = input('Enter the input sequence x(n):  ');
N = length(xn);
n=0:1:N-1;
k=0:1:N-1;

Xk = fft(xn,N);
disp('Xk: ');
disp(Xk);
mag_Xk = abs(Xk);
disp('magnitude values of Xk: ');
disp(Xk);
ang_Xk = angle(Xk);
disp('phase values of Xk: ');
disp(Xk);

subplot(3,1,1);
stem(n,xn);
title('input sequence x(n)')
xlabel('-----> n');
ylabel('-----> xn');


subplot(3,1,2);
stem(k,mag_Xk);
title('Magnitude response of X(k)');
xlabel('-----> k');
ylabel('-----> |Xk|');

subplot(3,1,3);
stem(k,ang_Xk);
title('Phase response of X(k)');
xlabel('-----> k');
ylabel('-----> phase of X(k)');
```

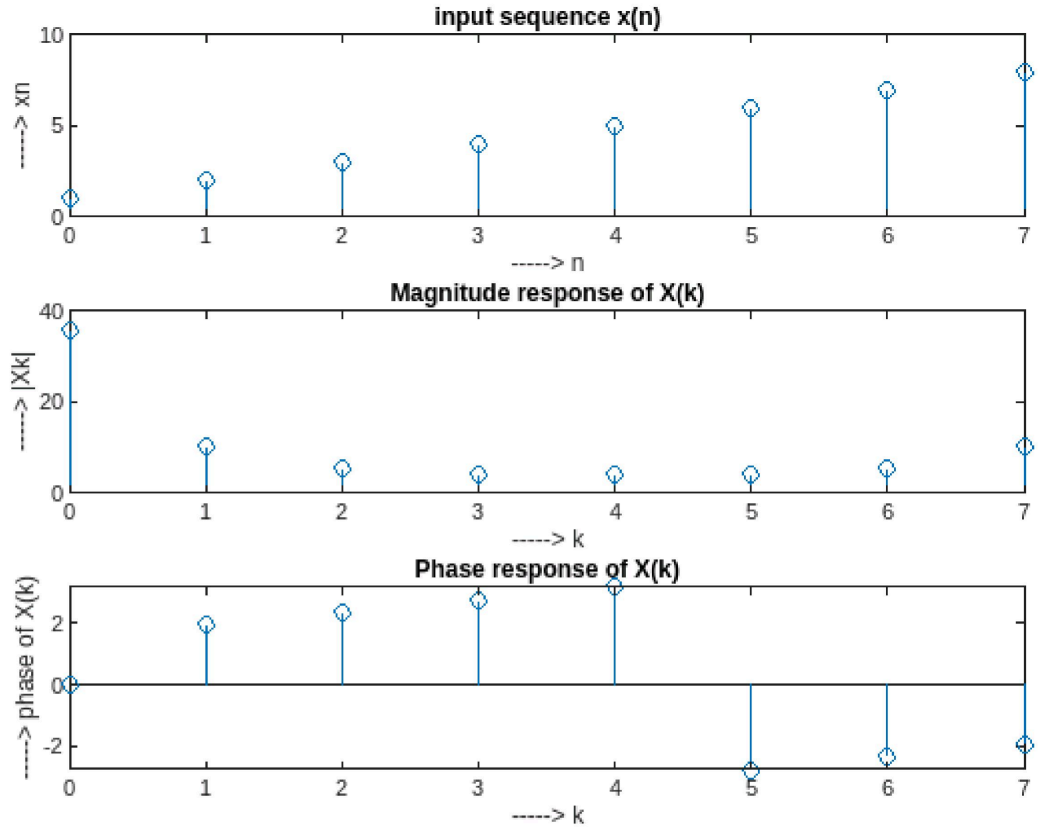
Result:

```
Enter the input sequence x(n):
[1 2 3 4 5 6 7 8]
X(k):
36.0000 + 0.0000i
-4.0000 + 9.6569i
-4.0000 + 4.0000i
-4.0000 + 1.6569i
-4.0000 + 0.0000i
-4.0000 - 1.6569i
```

-4.0000 - 4.0000i
-4.0000 - 9.6569i

magnitude values of Xk:
36.0000 10.4525 5.6569 4.3296 4.0000 4.3296 5.6569 10.4525
phase values of Xk:
0 1.9635 2.3562 2.7489 3.1416 -2.7489 -2.3562 -1.9635

Graph:



PROGRAM TITLE: **1b. Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum (Without Using inbuilt function).**

MATLAB CODE

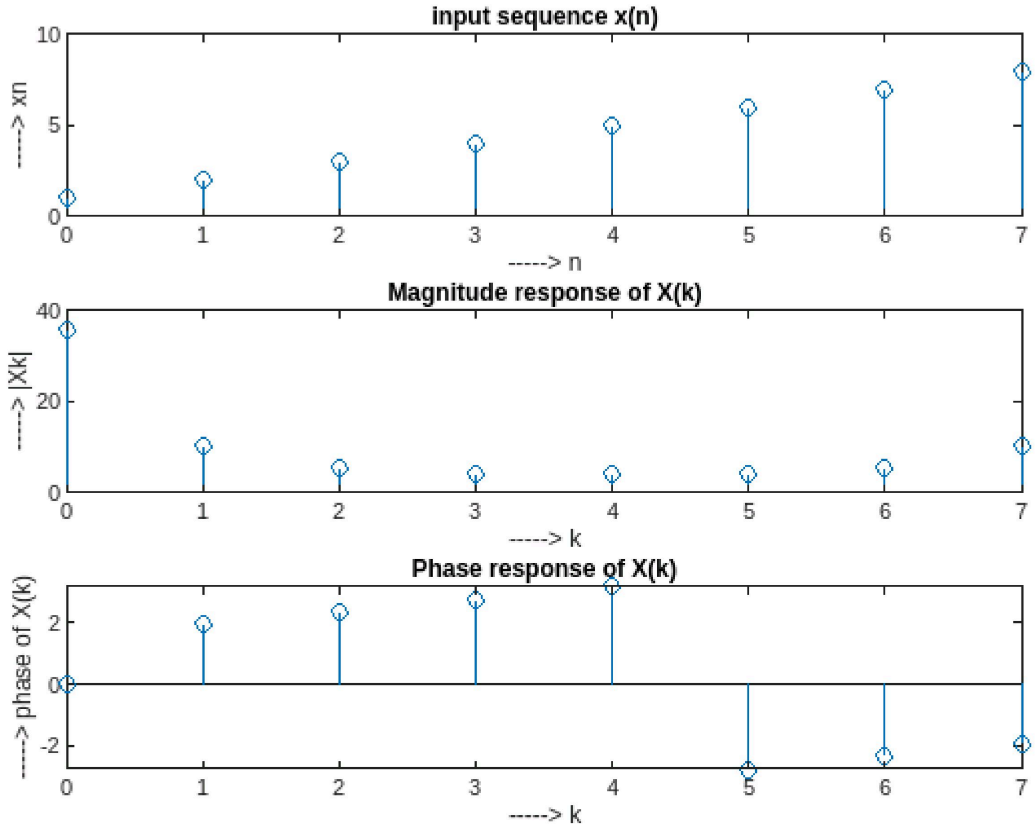
```
clc;
clear all;

xn = input('Enter the input sequence x(n):  ');
N = length(xn);

Xk = zeros(1,N);

for k=0:N-1
   for n=0:N-1
      Xk(k+1) = Xk(k+1)+xn(n+1)*exp((-i)*2*pi*k*n/N);
   end;
end


disp('X(k): ');
```

```
disp(Xk);
mag_Xk = abs(Xk);
disp('magnitude values of Xk: ');
disp(mag_Xk);
ang_Xk = angle(Xk);
disp('phase values of Xk: ');
disp(ang_Xk);

n=0:N-1;
k=0:N-1;
subplot(3,1,1);
stem(n,xn);
title('input sequence x(n)')
xlabel('-----> n');
ylabel('-----> xn');


subplot(3,1,2);
stem(k,mag_Xk);
title('Magnitude response of X(k)');
xlabel('-----> k');
ylabel('-----> |Xk|');

subplot(3,1,3);
stem(k,ang_Xk);
title('Phase response of X(k)');
xlabel('-----> k');
ylabel('-----> phase of X(k)');
```

Result:

```
Enter the input sequence x(n):
[1 2 3 4 5 6 7 8]
X(k):
36.0000 + 0.0000i
-4.0000 + 9.6569i
-4.0000 + 4.0000i
-4.0000 + 1.6569i
-4.0000 + 0.0000i
-4.0000 - 1.6569i
-4.0000 - 4.0000i
-4.0000 - 9.6569i

magnitude values of Xk:
36.0000  10.4525   5.6569   4.3296   4.0000   4.3296   5.6569  10.4525
phase values of Xk:
0   1.9635   2.3562   2.7489   3.1416  -2.7489  -2.3562  -1.9635
```

Graph:


input sequence x(n)
----> xn
----> n
Magnitude response of X(k)
----> |Xk|
----> k
Phase response of X(k)
----> phase of X(k)
----> k

PROGRAM TITLE: **2a. Matlab code to Linear and Circular convolution of given sequences.**

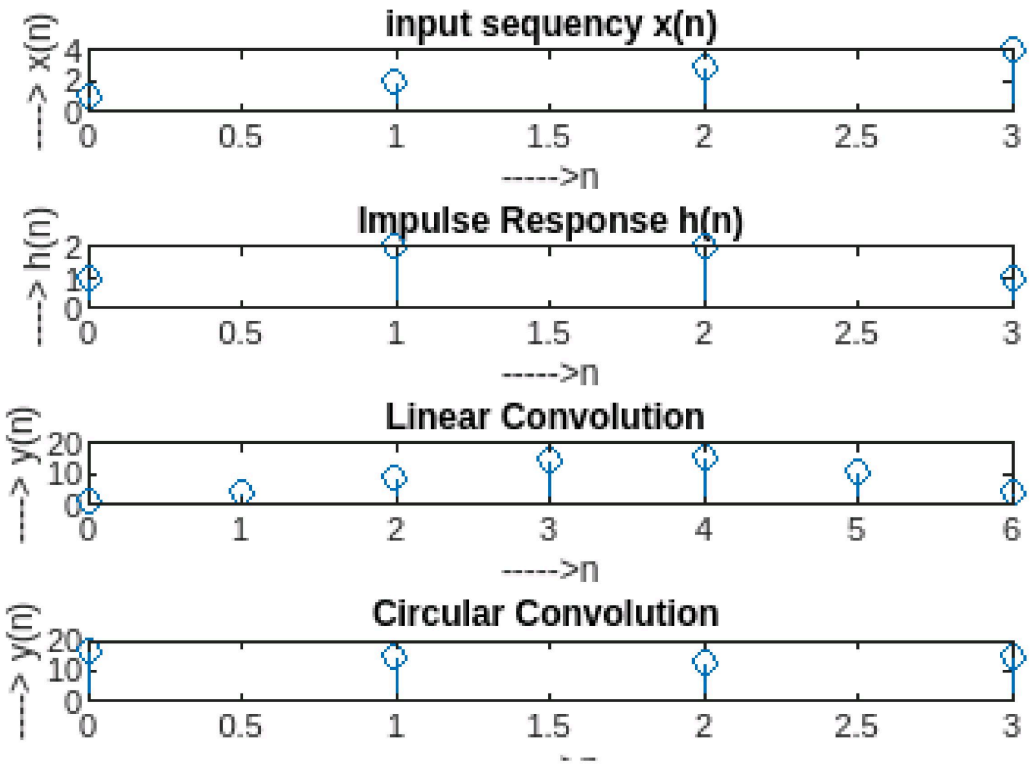
MATLAB CODE

```
clc
clear;

x=input('Enter the input signal x(n):  ');
h=input('Enter the impulse response h(n): ');
N = max(length(x),length(h));

y1=conv(x,h);
disp('The Linear convolution of x(n) and h(n) is : ')
disp(y1);

y2=cconv(x,h,N);
disp('The circular convolution of x(n) and h(n) is : ')
disp(y2);


subplot(4,1,1);
stem(0:length(x)-1, x);
title('input sequency x(n)');
xlabel('----->n');
ylabel('----> x(n)')


subplot(4,1,2);
stem(0:length(h)-1, h);
title('Impulse Response h(n)');
xlabel('----->n');
ylabel('----> h(n)')

subplot(4,1,3);
stem(0:length(y1)-1, y1);
title('Linear Convolution ');
xlabel('----->n');
ylabel('----> y(n)');

subplot(4,1,4);
stem(0:length(y2)-1, y2);
title('Circular Convolution ');
xlabel('----->n');
ylabel('----> y(n)');
```

Result:

```
Enter the input signal x(n):
[1 2 3 4]
Enter the impulse response h(n):
[1 2 2 1]
The Linear convolution of x(n) and h(n) is :
     1     4     9    15    16    11     4

The circular convolution of x(n) and h(n) is :
    17    15    13    15
```

Graph:


input sequency x(n)
---> x(n)
4
2
0
0
0.5
1
1.5
2
2.5
3
----->n
Impulse Response h(n)
---> h(n)
2
1
0
0
0.5
1
1.5
2
2.5
3
----->n
Linear Convolution
---> y(n)
20
10
0
0
1
2
3
4
5
6
----->n
Circular Convolution
---> y(n)
20
10
0
0
0.5
1
1.5
2
2.5
3

PROGRAM TITLE: **2a. Matlab code to Verify the Properties of Linear convolution.**

MATLAB CODE

```
clc
clear;

x=input('Enter the input signal x(n):  ');
h1=input('Enter the impulse response h1(n): ');
h2=input('Enter the impulse response h2(n): ');


%Commutative Property: x(n) * h(n) = h(n) * x(n)

LHS1 = conv(x,h1);
RHS1 = conv(h1,x);
disp('LHS:   ');
disp(LHS1);
disp('RHS:   ');
disp(RHS1);

if(LHS1 == RHS1)
    disp('Commutatuve Property Verified');
else
     disp('Commutatuve Property Not Verified');
end


%Distributive Property: x(n) * [h1(n)+h2(n)] = [x(n) * h1(n)] + [x(n) *
h2(n)]

LHS2 = conv(x,(h1+h2));
RHS2 = conv(x,h1) + conv(x,h2);
disp('LHS:   ');
disp(LHS2);
disp('RHS:   ');
disp(RHS2);

if(LHS2 == RHS2)
    disp('Distributive Property Verified');
else
     disp('Distributive Property Not Verified');
end

%Associative Property: [ x(n) * h1(n) ] * h2(n) = x(n) * [ h1(n)] * h2(n)]

LHS3 = conv(conv(x,h1),h2);
RHS3 = conv(x, conv(h1,h2));
disp('LHS:   ');
disp(LHS3);
disp('RHS:   ');
disp(RHS3);

if(LHS3 == RHS3)
    disp('Associative Property Verified');
else
     disp('Associative Property Not Verified');
end
```

Result:

```
Enter the input signal x(n):
[1 2 1]
Enter the impulse response h1(n):
[1 2 2]
Enter the impulse response h2(n):
[2 1 1]
LHS:
     1     4     7     6     2

RHS:
     1     4     7     6     2

Commutatuve Property Verified
LHS:
     3     9    12     9     3

RHS:
     3     9    12     9     3

Distributive Property Verified
LHS:
     2     9    19    23    17     8     2

RHS:
     2     9    19    23    17     8     2

Associative Property Verified
```

PROGRAM TITLE: **3. MATLAB PROGRAM TO FIND LINEAR CONVOLUTION USING DFT AND IDFT**

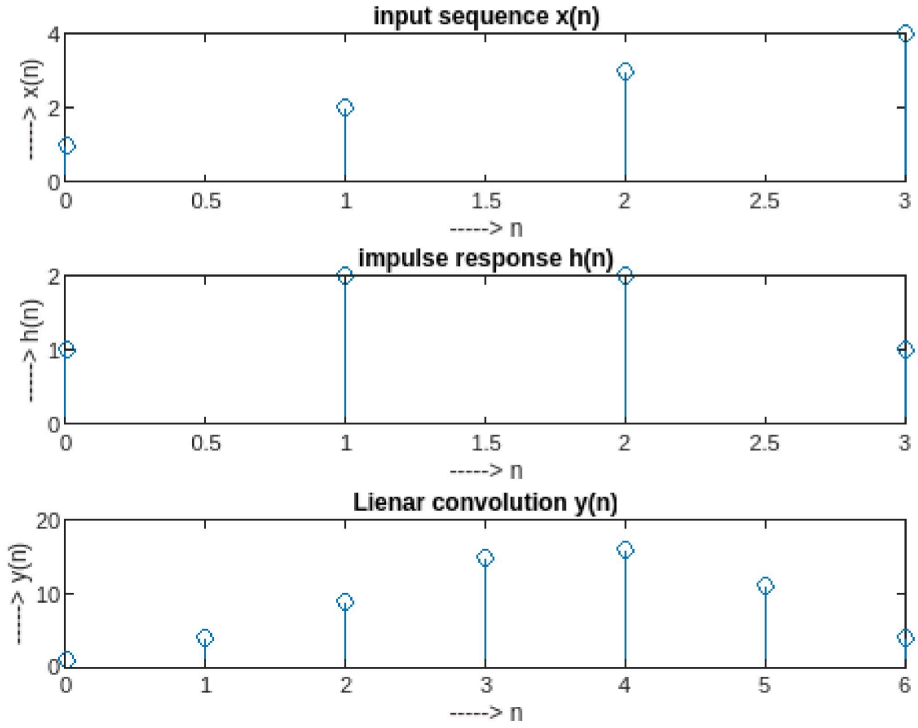
MATLAB CODE

```
clc;
clear all;

xn = input('Enter the input sequence x(n):  ');
hn = input('Enter the impulse response h(n): ');
N = length(xn)+length(hn)-1

Xk = fft(xn,N);
Hk = fft(hn,N);

Yk = Xk.* Hk

yn = ifft(Yk.');

disp('yn: ');
disp(yn);

subplot(3,1,1);
stem(0:length(xn)-1, xn);
xlabel('-----> n');
ylabel('-----> x(n)');
title('input sequence x(n)');

subplot(3,1,2);
stem(0:length(hn)-1, hn);
xlabel('-----> n');
ylabel('-----> h(n)');
title('impulse response h(n)');

subplot(3,1,3);
stem(0:length(yn)-1, yn);
xlabel('-----> n');
ylabel('-----> y(n)');
title('Lienar convolution y(n)');
```

Result:

```
Enter the input sequence x(n):
[1 2 3 4]
Enter the impulse response h(n):
[1 2 2 1]

yn:
    1.0000
    4.0000
    9.0000
   15.0000
   16.0000
   11.0000
    4.0000
```

Graph:


input sequence x(n)
----> x(n)
----> n
impulse response h(n)
----> h(n)
----> n
Lienar convolution y(n)
----> y(n)
----> n

PROGRAM TITLE: **4. MATLAB PROGRAM TO FIND CIRCULAR CONVOLUTION USING DFT AND IDFT**

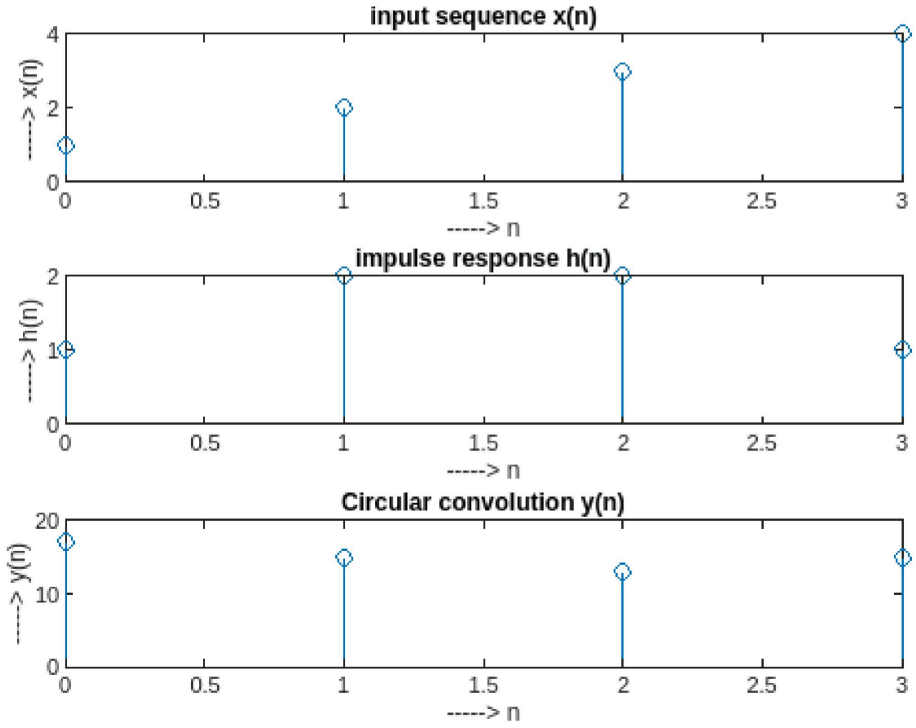
MATLAB CODE

```
clc;
clear all;

xn = input('Enter the input sequence x(n):  ');
hn = input('Enter the impulse response h(n): ');
N = max(length(xn),length(hn))

Xk = fft(xn,N);
Hk = fft(hn,N);

Yk = Xk.* Hk

yn = ifft(Yk.');

disp('yn: ');
disp(yn);

subplot(3,1,1);
stem(0:length(xn)-1, xn);
xlabel('-----> n');
ylabel('-----> x(n)');
title('input sequence x(n)');

subplot(3,1,2);
stem(0:length(hn)-1, hn);
xlabel('-----> n');
ylabel('-----> h(n)');
title('impulse response h(n)');

subplot(3,1,3);
stem(0:length(yn)-1, yn);
xlabel('-----> n');
ylabel('-----> y(n)');
title('Circular convolution y(n)');
```

Result:

```
Enter the input sequence x(n):
[1 2 3 4]
Enter the impulse response h(n):
[1 2 2 1]

yn:
    17
    15
    13
    15
```

Graph:


input sequence x(n)
----> x(n)
----> n
impulse response h(n)
----> h(n)
----> n
Circular convolution y(n)
----> y(n)
----> n

PROGRAM TITLE: **5A. MATLAB PROGRAM FOR linearity Property Verification**
DFT{ a1*x1(n) + a2*x2(n) } = a1*X1(k) + a2*X2(k)

MATLAB CODE

```
clc;
clear all;

x1n = input('x1(n):  ' );
x2n = input('x2(n):  ' );
N = max(length(x1n), length(x2n));
a1 = input('a1:  ' );
a2 = input('a2:  ' );


% code for DFT{ a1*x1(n) + a2*x2(n) }
yn = a1*x1n + a2*x2n;
LHS = fft(yn,N);
disp('LHS : ')
disp(LHS.');


% code for a1*X1(k) + a2*X2(k)
X1k = fft(x1n,N);
X2k = fft(x2n,N);
RHS = a1*X1k + a2*X2k;
disp('RHS :')
disp(RHS.');

if LHS == RHS
    disp('Lineartiy Property Verified');
else
    disp('Lineartiy Property not Verified');

end
```

Result:

```
Enter the input sequence x(n):
[1 2 3 4]
Enter the impulse response h(n):
[1 2 2 1]

yn:
    17
    15
    13
    15
```

PROGRAM TITLE: **5B. Matlab Program for Circular Time Shift Property DFT(x((n-m))N) = e(-j*2*pi*m*k)*X(k)**

MATLAB CODE

```
clc;
clear all;

xn = input('xn:   ');
N = length(xn);
m = 4;
yn = circshift(xn,[0,m]);


LHS = int32(fft(yn,N));
disp('LHS: ');
disp(LHS.');


Xk = fft(xn,N);
RHS = zeros(1,N);
for k=0:N-1
    RHS(k+1) =int32(exp(-2*pi*i*m*k/N)*Xk(k+1));
end
disp('RHS');
disp(RHS.');

if LHS == RHS
    disp('The circular Time Shift Property verified');
else
    disp('The circular Time Shift property not Verified')

end
```

Result:

```
xn:
[1 2 3 4]
LHS:
    10 +     0i
    -2 +     2i
    -2 +     0i
    -2 -     2i

RHS
  10.0000 + 0.0000i
  -2.0000 + 2.0000i
  -2.0000 + 0.0000i
  -2.0000 - 2.0000i

The circular Time Shift Property verified
```

PROGRAM TITLE: **5C. Matlab Program for Circular Frequency Shift Property**
DFT{e(j*2*pi*m*n)*x(n)} = X((k-m))N

MATLAB CODE

```
%Circular freq shift Property
%DFT{e(j*2*pi*m*n)*x(n)} = X((k-m))N

clc;
clear all;

xn = input('xn:   ');
N = length(xn)
m = 2
yn = zeros(1,N)
for n=0:N-1
    yn(n+1)=exp(2*pi*i*n*m/N)*xn(n+1);
end
LHS = int32(fft(yn,N));
disp('LHS: ');
disp(LHS.');



RHS = fft(xn,N)
RHS = int32(circshift(RHS,[0 m]));
disp('RHS: ');
disp(RHS.');



if LHS == RHS
    disp('The circular Frequecny Shift Property verified');
else
    disp('The circular Frequency Shift property not Verified')

end
```

Result:

```
xn:
[1 2 3 4]

LHS:
    -2 +     0i
    -2 -     2i
    10 +     0i
    -2 +     2i


RHS =

  10.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 -
2.0000i

RHS:
    -2 +     0i
    -2 -     2i
    10 +     0i
    -2 +     2i
```

```
The circular Frequecny Shift Property verified
```

PROGRAM TITLE: **6. Matlab Program for Parsvels theorm verification (Energy Verification)**

MATLAB CODE

```
clc;
clear all;

xn = input('xn: ');
N = length(xn);

LHS = 0;
for n=0:N-1
    LHS = LHS + (abs(xn(n+1)).*abs(xn(n+1)));
end

RHS = 0;
Xk = fft(xn,N);
for k=0:N-1
    RHS = RHS + (abs(Xk(k+1)).*abs(Xk(k+1)));
end
RHS = RHS/N;

disp('RHS:');
disp(RHS);
disp('LHS:');
disp(LHS);

if LHS == RHS
    disp('The Parsvels Theorm verified');
else
    disp('The Parsvels Theorm not Verified')
```

Result:

```
[1 2 3 4]
RHS:
    30

LHS:
    30

The Parsvels Theorm verified
```

PROGRAM TITLE: **7. Design and implementation of IIR (Butterworth)low pass filter to meet given specifications.**
Passband attenuation ≤1.25db,stopband attenuation ≥ 15db fpass=200Hz, fstop=300Hz fsample=2KHz

MATLAB CODE

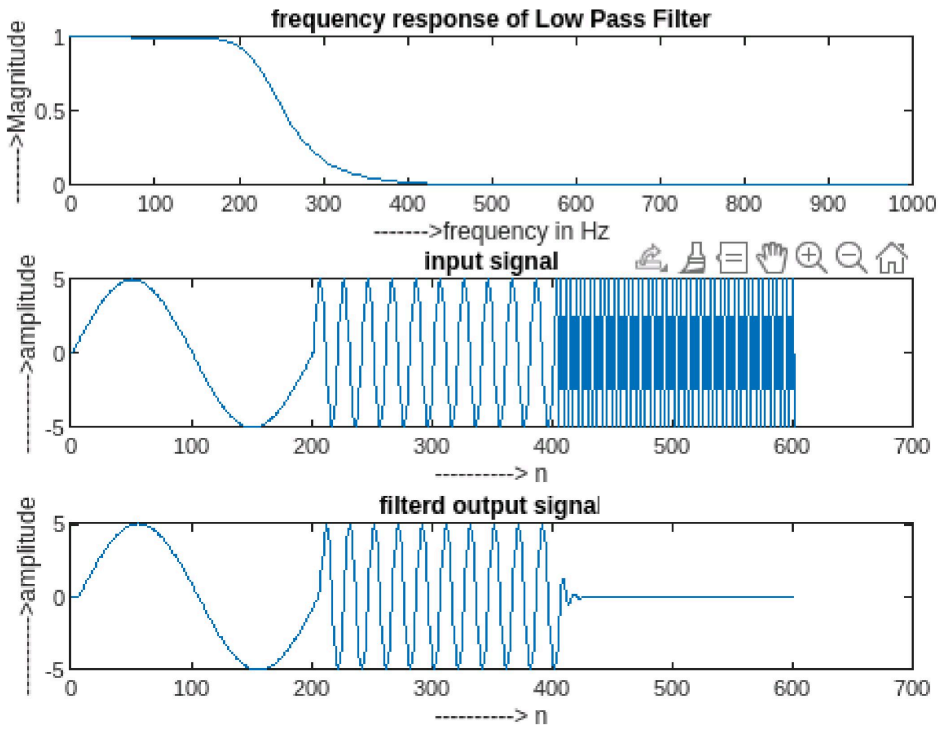
```
clc;
clear all;

%given specifications
Ap=1.25;As=15;
fpb=200; fsb=300; fs = 2000;

%to find Order(N) and cutoff frequency (fc)
fpbn = fpb /(fs/2); fsbn = fsb /(fs/2);

[N,fc]=buttord(fpbn,fsbn,Ap,As);
disp('order of the filter is =');
disp(N);
disp('cutoff frequency is = ');
disp(fc*fs/2);

%to compute frequency response of an IIR digital filter
[b,a]=butter(N,fc);
[H,f] = freqz(b,a,256,fs);

subplot(3,1,1);
plot(f,abs(H));
title('frequency response of Low Pass Filter');
xlabel('------->frequency in Hz');
ylabel('------->Magnitude');



%filtering operation on input singal having frequency 10Hz,100Hz,500Hz
n = 0:1/fs:0.1;
s1=5*sin(2*pi*10*n);
s2=5*sin(2*pi*100*n);
s3=5*sin(2*pi*500*n);

x = [s1 s2 s3];
subplot(3,1,2);
plot(x);
title('input signal');
xlabel('----------> n');
ylabel('---------->amplitude');

y = filter(b,a,x);
subplot(3,1,3); plot(y);
title('filterd output signal');
xlabel('----------> n');
ylabel('---------->amplitude')
```

Result:

```
order of the filter is =
     6
```

```
cutoff frequency is =
  232.9175
```

Graph:



PROGRAM TITLE: **8. Design and implementation of IIR (Butterworth)High pass filter to meet given specifications.*Passband attenuation ≤1.25db,stopband attenuation ≥ 15db fpass=300Hz, fstop=200Hz fsample=2KHz***

MATLAB CODE

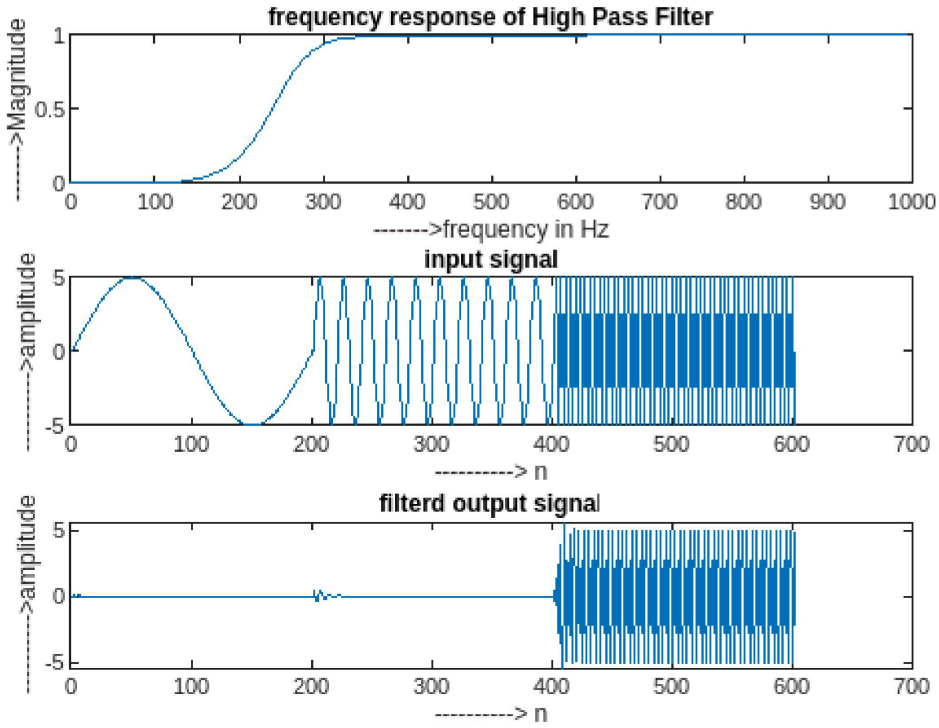
```
clc;
clear all;

%given specifications
Ap=1.25;As=15;
fpb=300; fsb=200; fs = 2000;

%to find Order(N) and cutoff frequency (fc)
fpbn = fpb /(fs/2); fsbn = fsb /(fs/2);

[N,fc]=buttord(fpbn,fsbn,Ap,As);
disp('order of the filter is =');
disp(N);
disp('cutoff frequency is = ');
disp(fc*fs/2);

%to compute frequency response of an IIR digital filter
[b,a]=butter(N,fc,'high');
[H,f] = freqz(b,a,256,fs);

subplot(3,1,1);
plot(f,abs(H));
title('frequency response of High Pass Filter');
xlabel('------->frequency in Hz');
ylabel('------->Magnitude');



%filtering operation on input singal having frequency 10Hz,100Hz,500Hz
n = 0:1/fs:0.1;
s1=5*sin(2*pi*10*n);
s2=5*sin(2*pi*100*n);
s3=5*sin(2*pi*500*n);

x = [s1 s2 s3];
subplot(3,1,2);
plot(x);
title('input signal');
xlabel('----------> n');
ylabel('---------->amplitude');

y = filter(b,a,x);
subplot(3,1,3); plot(y);
title('filterd output signal');
xlabel('----------> n');
ylabel('---------->amplitude')
```

Result:

```
order of the filter is =
     6
```

```
cutoff frequency is =
  259.6729
```

Graph:



PROGRAM TITLE: **9. Design and implementation of FIR Low pass filter to meet given specifications.** ***fpass=100Hz, fstop=200Hz fsample=2KHz, hamming window***

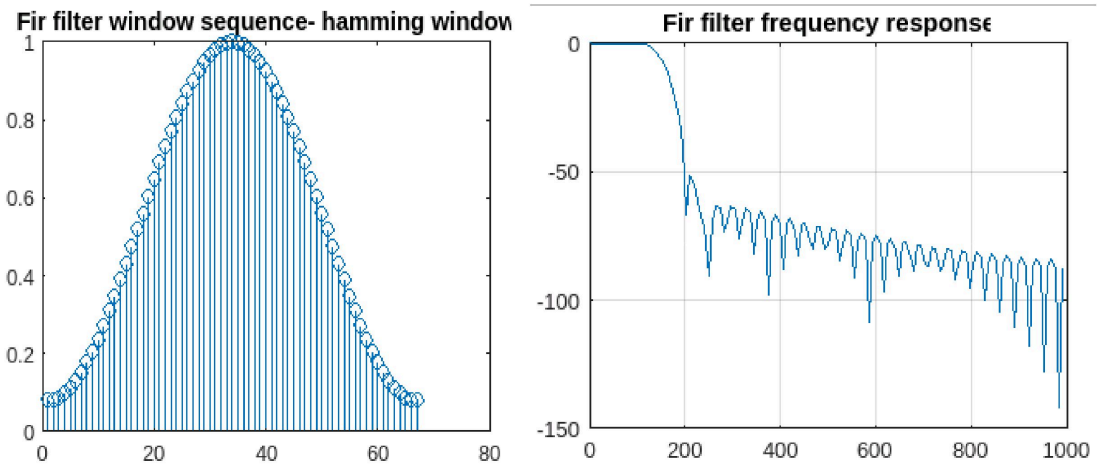
MATLAB CODE

```
clc;
clear all;
wpa=input('Enter passband edge frequency in Hz');
wsa= input('Enter stopband edge frequency in Hz');
ws1= input('Enter sampling frequency in Hz');

%Calculate transmission BW,Transition band tb,order of the filter
wpd=2*pi*wpa/ws1;
wsd=2*pi*wsa/ws1;
tb=wsd-wpd;
N=ceil(6.6*pi/tb) ;
wc=(wsd+wpd)/2;

%compute the normalized cut off frequency
wc=wc/pi;

%calculate & plot the window
hw=hamming(N+1);
stem(hw);
title('Fir filter window sequence- hamming window');

%find h(n) using FIR
h=fir1(N,wc,hamming(N+1));

%plot the frequency response
figure(2);
[m,w]=freqz(h,1,128);
mag=20*log10(abs(m));
plot(ws1*w/(2*pi),mag);
title('Fir filter frequency response');
grid;
```

Result:

```
Enter passband edge frequency in Hz
100
Enter stopband edge frequency in Hz
200
Enter sampling frequency in Hz
2000
```

Graph:



PROGRAM TITLE: **10. Design and implementation of FIR High pass filter to meet given specifications.** ***fpass=200Hz, fstop=100Hz fsample=2KHz, hamming window***

MATLAB CODE

```
clc;
clear all;
wpa=input('Enter passband edge frequency in Hz');
wsa= input('Enter stopband edge frequency in Hz');
ws1= input('Enter sampling frequency in Hz');

%Calculate transmission BW,Transition band tb,order of the filter
wpd=2*pi*wpa/ws1;
wsd=2*pi*wsa/ws1;
tb=wpd-wsd;
N=ceil(6.6*pi/tb) ;
wc=(wsd+wpd)/2;

%compute the normalized cut off frequency
wc=wc/pi;

%calculate & plot the window
hw=hamming(N+1);
stem(hw);
title('Fir filter window sequence- hamming window');

%find h(n) using FIR
h=fir1(N,wc,'high');

%plot the frequency response
figure(2);
[m,w]=freqz(h,1,128);
mag=20*log10(abs(m));
plot(ws1*w/(2*pi),mag);
title('Fir filter frequency response');
grid;
```

Result:

```
Enter passband edge frequency in Hz
200
Enter stopband edge frequency in Hz
100
Enter sampling frequency in Hz
2000
```

Graph: