Q1)Method-1: Compute DFT and IDFT using inbuilt numpy function (fft, ifft) in python import numpy as np

from numpy.fft import fft, ifft #N-4 point DFT and IDFT

#input sequence x-(1,3,5,2]

L-len(x) #print (L)

Nint(input("\n Enter number of DFT points: \n"))

m-N-L #Trailing zeros x1-np.pad(x, (0,m), 'constant)

print("in zero padded sequence: \n",x1)

#compute DFT Xk-fft(x1,N) #display the result

print("in DFT of the given sequence is: \n",Xk)

#compute IDFT

xN-ifft(Xk,N)

xNR-np.round(xN.real, 1) print("\n IDFT is: \n",xNR)

Output:

Enter number of DFT points:

4

zero padded sequence: [1 3 5 2]

DFT of the given sequence is: [11.+0.1-4.-1.1) 1.+0.1 4.+1.11

IDFT is: [1. 3. 5. 2.1
****

Q2)To plot Magnitude and Phase Spetrum of N point DFT

```
import numpy as np
from numpy.fft import fft, ifft #input sequence
x=[2,3,4,2]
L= len(x)
#print (L)
N=int(input("Enter number of DFT points: "))
m-N-L #Trailing zeros #Zero padded sequence
x1=np.pad(x, (0,m), 'constant)
print("zero padded sequence: ",x1)
#compute N-point DFT Xk=fft(x1,N)
#compute N-point IDFT
XN=ifft (Xk,N)
XNR-np.round(xN.real,1)
```

```
print("IDFT is: ",xNR)
#plot magnitude and phase response of DFT from matplotlib import pyplot as plt
n=np.arange(N)
 #Discrete time range
k=np.arange(N)
#Discrete frequency index
plt.stem(n,x1)
plt.xlabel("--->n")
plt.ylabel("Amplitude")
plt.title("input sequence")
plt.show()
#magnitude spectrum Xmag-np.abs (Xk)
plt.stem (k, Xmag)
plt.xlabel("-->k")
plt.ylabel("Magnitude")
plt.title("Magnitude
plt.show()
#phase spectrum
Xph-np.angle(Xk)
plt.stem(k, Xph)
plt.xlabel("--->k")
plt.ylabel("Phase")
plt.title("Phase response")
plt.show()
```

****

Q3)Method 3 Linear Convolution using equation

```
import numpy as np
x-eval(input("Enter sequnce x[n] : "))
heval (input("Enter sequnce h[n] : ” ))
N * 1 = len(x)
 N * 2 = len(h)
N =N1+N2-1
m = N – N1
 n = N - N2
x = np.pad ( x, (0, m) ,"constant")
h = np.pad(h, (0, n) ,"constant")
y = np (N)
print("\nzero padded input sequence:\n",x)
print("\nzero padded input sequence:\n",h)
#print(y)
for n in range (N): for k in range (N):
if n>=k:
y[n]=y[n]+x[n-k]*h[k]
print("\n Result of Linear Convolution\n", y)
```

****

Q4)MINIMUM MAX MIXED PHASE SYSTEMS

```
#import pockages
```

import numpy as np from matplotlib import pyplot as plt from scipy import signal

Buser defined function for pole-zero plot def

polezero(b,a): (zeros, poles,gain)-signal. tf2zpk(b,a)

angle np.linspace (0,2*np.pt, 100) cirx-np.sin(angle)

ciry np.cos(angle)

plt.figure() plt.grid()

#Plot unit circle

plt.plot(cirx,ctry, 'k-') #plot poles and zeros in z plane

plt.plot(poles.real, poles.imag, 'bx'', zeros.real, zeros.imag, 'ro') plt.xlim(-3,3)

plt.xlabel('Real of 2') plt.ylabel('Imag of z')

plt.title('Pole-zero plot')

plt.ylim(-3,3) return(zeros,poles,gain)

#Define H(Z)

#Num coffe of H(Z)

b = [1, 0.5, 1.5] #Deno coff of H(Z) #call the function polezero (b, a)

a = [1, 0, 0]

#Display poles and zeros (zeros,poles,gain)-signal. tf2zpk(b,a)

print('in zero of H(Z):\n',zeros) print(in poles of H(Z):\n',poles) print('In gain of H(Z) : backslash n^ prime ,ga(n)

b * 1 = [1, 1, 1/6] a * 1 = [1, 0, theta]

#call the function polezero(bi,a1)

#Display poles and zeros (zeros,poles,gain) signal.tf2zpk(b1,a1)

print('in zero of H(Z) : backslash n^ * ,zeros) print('in poles of H(Z) : backslash n^ prime ,poles) print('in gain of H(Z) : backslash n^ prime ,gain)

b * 2 = [1, 2, 3] a * 2 = [1, theta, theta]

#call the function

polezero o (b2,a2)

#Display poles and zeros (zeros,poles,gain)-signal.tf2zpk(b2a2)

print('in zero of H(Z) : backslash n^ prime zeros) print('in poles of H(Z) :ln^ i ,poles)
plt.show()

****
Q5) Call the function to display pole zero

plt.figure(3)
plt.subplot(1,2,1)

polezero(num2, den2)

plt.grid()

plt.title('Type 2-pole zero') #plot phase response

(w,H) signal. freqz (num2, den2)
plt.subplot(1,2,2)
plt.phase spectrum(H)

plt.grid(

plt.title(Type2 Phase Char')

plt.show()

#TYPE3

#Define H(Z)

#NUM coeff (Z)
num3-[-2.5,3,0,-3,2.5]
den3 [1,0,0,0,0]

#call the function to display pole-zero

plt.figure(4)

plt.subplot(1,2,1) polezero(num3, den3)

plt.grid()

plt.title(Type 3-pole zero')

#plot phase response

(w,H)=signal.freqz (num3, den3)

plt.subplot(1,2,2)
plt.phase spectrum(H)
plt.grid()

```
plt.title('Type3 Phase Char')

plt.show() #TYPE4

#Define H(Z)

#NUM coeff H(Z)
num4-[1.2,-2.5,2.5, -1.2]

den4=[1,0,0,0]

#call the function to display pole-zero

plt.figure(5)

plt.subplot(1,2,1)

polezero(num4, den4)

plt.grid()

plt.title('Type 4-pole zero')

#plot phase response
(w,H)=signal.freqz (num4,den4)

plt.subplot(1,2,2)

plt.phase spectrum(H)

plt.grid()
plt.title(Type4 Phase Char')
print("\n gain of H(Z): \n',gain)

b3=[ 1 ,0.5,1.5 ]

a4= [1,0,0 ]
#coll the function
 polezero o (b3,a4)

#Display poles and zeros
 (zeros, poles,gain)=signal.tf2zpk(b,a)

print('\n zero of H(Z): \n',zeros)

print('\n poles of H(Z) : backslash r ',poles)
print('n gain of H(Z):\n',gain)
```