1) Write python program to compute DFT and IDFT (without using inbuilt function) of 4- point sequence, x[n]={ 2,4, 1,3}. Verify your answers by calculation.
What is twiddle factor? What are its properties?

Code -

DFT
```
import numpy as np
def DFT(x):
 N=len(x)
 n=np.arange(N)
 print('\n Time index:',n)
 K=np.arange(N)
 K=K.reshape(N,1)
 print('\nTranspose of frequency index k:\n',K)
 W=np.exp(-2j*np.pi*n*K/N)
 Wt=np.round(W.real,1)+np.round(W.imag,1)*1j
 print('\n Twiddle factor matrix:\n',Wt)
 return(np.dot(Wt,x))
x1=[1,2,3,4]
x1K=DFT(x1)
print('X(n):',x1)
print('X(k):',x1K)
```

IDFT
```
import numpy as np
def IDFT(x):
 N=len(x)
 n=np.arange(N)
 print('\n Time index:',n)
 K=np.arange(N)
 K=K.reshape(N,1)
 print('\nTranspose of frequency index k:\n',K)
 W=np.exp(2j*np.pi*n*K/N)
 Wt=np.round(W.real,1)+np.round(W.imag,1)*1j
 print('\n Twiddle factor matrix:\n',Wt)
 return(np.dot(Wt,x)/N)
x1=[10,-2+2j,-2,-2-2j]
x1n=IDFT(x1)
print('X(n):',x1)
print('x(n):',x1n)
```

2.)Write python program to plot window functions in Time Domain and its magnitude spectrum.
Also display window coefficients and verify the same by calculation.

code-
```
import numpy as np
from matplotlib import pyplot as plt
from scipy import signal
```

```
# Define window parameter
N = 11 # Length of the window
n = np.arange(N)
# Rectangular window
win1 = np.ones(N)
print("Rectangular window:")
print(win1)
plt.subplot(5, 2, 1)
plt.stem(n, win1)
plt.title('Rectangular Window')
plt.xlabel('Time Range')
plt.ylabel('Weight')
a = 1
w, H = signal.freqz(win1, a)
Hm = np.abs(H)
Hdb = 20 * np.log10(Hm)
plt.subplot(5, 2, 2)
plt.plot(w / max(w), Hdb)
plt.title('Magnitude in dB (Rectangular)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Magnitude (dB)')
plt.grid()

# Hamming window
win2 = signal.hamming(N)
print("Hamming window:")
print(win2)
plt.subplot(5, 2, 3)
plt.stem(n, win2)
plt.title('Hamming Window')
plt.xlabel('Time Range')
plt.ylabel('Weight')
w, H = signal.freqz(win2, a)
Hm = np.abs(H)
Hdb = 20 * np.log10(Hm)
plt.subplot(5, 2, 4)
plt.plot(w / max(w), Hdb)
plt.title('Magnitude in dB (Hamming)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Magnitude (dB)')
plt.grid()

# Hanning window
win3 = signal.hann(N)
print("Hanning window:")
print(win3)
plt.subplot(5, 2, 5)
plt.stem(n, win3)
plt.title('Hanning Window')
plt.xlabel('Time Range')
plt.ylabel('Weight')
w, H = signal.freqz(win3, a)
Hm = np.abs(H)
Hdb = 20 * np.log10(Hm)
```

```
plt.subplot(5, 2, 6)
plt.plot(w / max(w), Hdb)
plt.title('Magnitude in dB (Hanning)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Magnitude (dB)')
plt.grid()

# Blackman window
win4 = signal.blackman(N)
print("Blackman window:")
print(win4)
plt.subplot(5, 2, 7)
plt.stem(n, win4)
plt.title('Blackman Window')
plt.xlabel('Time Range')
plt.ylabel('Weight')
w, H = signal.freqz(win4, a)
Hm = np.abs(H)
Hdb = 20 * np.log10(Hm)
plt.subplot(5, 2, 8)
plt.plot(w / max(w), Hdb)
plt.title('Magnitude in dB (Blackman)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Magnitude (dB)')
plt.grid()

# Bartlett window
win5 = signal.bartlett(N)
print("Bartlett window:")
print(win5)
plt.subplot(5, 2, 9)
plt.stem(n, win5)
plt.title('Bartlett Window')
plt.xlabel('Time Range')
plt.ylabel('Weight')
w, H = signal.freqz(win5, a)
Hm = np.abs(H)
Hdb = 20 * np.log10(Hm)
plt.subplot(5, 2, 10)
plt.plot(w / max(w), Hdb)
plt.title('Magnitude in dB (Bartlett)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Magnitude (dB)')
plt.grid()
plt.show()
```

3) Write python program to plot pole zero diagram of following system. Identify the systems as minimum/maximum/mixed phase.
(A transfer function will be given at the time of exam)

Code-

```
import numpy as np
from matplotlib import pyplot as plt
a=[6,1,-1]
b=[1,0,0]
zeros=np.roots(a)
poles=np.roots(b)
print("zeros=",zeros)
print("poles=",poles)
if (np.all(np.abs(zeros)<1)):
 print("System is Minimum Phase FIR System")
elif(np.all(np.abs(zeros)>1)):
 print("System is Maximum Phase FIR System")
else:
 print("System is Mixed Phase FIR System")

plt.figure()
plt.scatter(np.real(zeros),np.imag(zeros),color='red',
marker='o',label='Zeros')
plt.scatter(np.real(poles),np.imag(poles),color='blue'
,marker='x',label='Poles')
unit_circle=plt.Circle((0,0),1,color='black',fill=Fals
e,linestyle='--',linewidth=1)
plt.gca().add_artist(unit_circle)
plt.axvline(0,color='black',linewidth=0.5)
plt.axhline(0,color='black',linewidth=0.5)
plt.xlim(-2,2)
plt.ylim(-2,2)
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.title('Pole zero diagram with unit circle')
plt.grid(True)
plt.legend()
plt.gca().set_aspect('equal',adjustable='box')
plt.show()
inside_unit_circle=np.all(np.abs(zeros)<1)
outside_unit_circle=np.all(np.abs(zeros)>1)
if inside_unit_circle:
 phase_type="Minimum Phase System"
elif outside_unit_circle:
 phase_type="Maximum Phase System"
else:
 phase_type="Mixed Phase System"
print("The system is a",phase_type)
```

4) Write python program to design FIR HPF with cutoff frequency pi/2 and length N=11 using rectangular and Hanning window.
Plot the designed filter characteristics.
Verify the filter coefficients (h[n]) by calculation.

Code-

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from scipy.signal import freqz, get_window

# Parameters
N = 11                  # Filter length
cutoff_freq = np.pi / 2  # Cutoff frequency in radians

# Generate the ideal high-pass filter (sinc function
in frequency domain)
n = np.arange(0, N)
M = (N - 1) / 2
h_ideal = np.sinc((n - M) * (cutoff_freq / np.pi))
h_ideal *= np.cos((n - M) * cutoff_freq)  # Convert to
HPF by cos modulation
h_ideal[(N-1)//2] = 1 - (cutoff_freq / np.pi)  # Avoid
NaN at M

# Window functions
rectangular_window = np.ones(N)
hanning_window = get_window('hann', N)

# Apply windows to the ideal HPF
h_rectangular = h_ideal * rectangular_window
h_hanning = h_ideal * hanning_window

# Frequency response
w, h_rect = freqz(h_rectangular, 1)
_, h_hann = freqz(h_hanning, 1)

# Plot filter coefficients
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.stem(n, h_rectangular, 'b', markerfmt='bo',
basefmt=" ", label="Rectangular Window")
plt.stem(n, h_hanning, 'r', markerfmt='ro', basefmt="
", label="Hanning Window")
plt.title("Filter Coefficients (h[n])")
plt.xlabel("n")
plt.ylabel("h[n]")
plt.legend()
plt.grid()

# Plot frequency response
plt.subplot(2, 1, 2)
plt.plot(w / np.pi, 20 * np.log10(abs(h_rect)), 'b',
label="Rectangular Window")
plt.plot(w / np.pi, 20 * np.log10(abs(h_hann)), 'r',
label="Hanning Window")
plt.title("Frequency Response of Designed FIR HPF")
plt.xlabel("Normalized Frequency (×π rad/sample)")
plt.ylabel("Magnitude (dB)")
plt.legend()
plt.grid()

plt.tight_layout()
```

```
plt.show()
```

5) Design a FIR LPF using pole zero placement method. Write python program to plot pole zero diagram and magnitude response of the same.

code -

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Coefficients for the filter
a = [1, 0.8]   # Numerator coefficients
b = [1, 0]     # Denominator coefficients

# Calculate zeros and poles
zeros = np.roots(a)
poles = np.roots(b)

# Plot Pole-Zero Diagram
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.scatter(np.real(zeros), np.imag(zeros),
color='red', marker='o', label='Zeros')
plt.scatter(np.real(poles), np.imag(poles),
color='blue', marker='x', label='Poles')
theta = np.linspace(0, 2 * np.pi, 100)
plt.plot(np.cos(theta), np.sin(theta), linestyle='--',
color='black')
plt.axvline(0, color='black', linewidth=0.5)
plt.axhline(0, color='black', linewidth=0.5)
plt.title('Pole-Zero Diagram')
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.grid(True)
plt.legend()

# Plot Magnitude Response
w, h = freqz(a, b)
plt.subplot(1, 2, 2)
plt.plot(w / np.pi, np.abs(h))
plt.title('Magnitude Response')
plt.xlabel('Normalized Frequency (×π rad/sample)')
plt.ylabel('Magnitude')
plt.grid(True)

plt.tight_layout()
plt.show()
```

6) Design IIR NOTCH filter using polezero placement method.
Write a python program to plot pole zero diagram and magnitude response of the same.

code - same as code number 5

7) Write python program to plot window functions in Time Domain and its magnitude spectrum.
Also display window coefficients and verify the same by calculation

code- same as code 2

8) Write python program to design FIR LPF with cutoff frequency pi/2 and length N=11 using Rectangular and Hanning window.
Plot the designed filter characteristics.
Verify the filter coefficients (h[n]) by calculation

code - same as code number 4

9) Write python program to compute DFT and IDFT (without using inbuilt function) of 4- point sequence, x[n]={ 2,5,1,3}.
Verify your answers by calculation.
What is twiddle factor? What are its properties?

code- same as code number 1

10) Write python program to plot magnitude spectrum of DFT of 4- point sequence, x[n]={ 2,5,1,3}.
Verify your answers by calculation.
Also plot the magnitude spectrum for N=8, 16 and 32
What is the effect of increase in value of N on DFT spectrum and IDFT?

```
import numpy as np
import matplotlib.pyplot as plt
```

code-

```
# Define the sequence
x = np.array([2, 5, 1, 3])
N = len(x)

# Calculate DFT manually for N=4
def manual_dft(x):
    N = len(x)
    X = np.zeros(N, dtype=complex)
    for k in range(N):
        for n in range(N):
```

```
            X[k] += x[n] * np.exp(-2j * np.pi * k * n
/ N)
    return X

# DFT for N=4
X_dft_4 = manual_dft(x)

# Function to compute and plot magnitude spectrum
def plot_magnitude_spectrum(X, N, title):
    plt.figure(figsize=(10, 5))
    plt.stem(np.arange(N), np.abs(X),
use_line_collection=True)
    plt.title(title)
    plt.xlabel('Frequency index (k)')
    plt.ylabel('Magnitude')
    plt.grid()
    plt.xticks(np.arange(N))
    plt.show()

# Plot DFT for N=4
plot_magnitude_spectrum(X_dft_4, N, "Magnitude
Spectrum of DFT for N=4")

# Now let's compute DFT for N=8, N=16, N=32 with zero-
padding
for new_N in [8, 16, 32]:
    x_padded = np.pad(x, (0, new_N - N), 'constant')
    X_dft = np.fft.fft(x_padded)
    plot_magnitude_spectrum(X_dft, new_N, f"Magnitude
Spectrum of DFT for N={new_N}")

# Print the DFT values for verification
print("DFT for N=4 (manual calculation):")
print(X_dft_4)
print("\nMagnitude spectrum for N=4:")
print(np.abs(X_dft_4))
```

11) Write a python program to synthesize a single sinusoidal signal consisting of frequency components 15 Hz and 50 Hz. Filter this signal using a notch filter having a notch frequency of 50 Hz. Display the original signal, magnitude response of notch filter and filtered signal. (EXP 10)

Code-

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import signal
f1 = 15
f2 = 50
n = np.linspace(0, 1, 1000)
```

```
noisysignal = np.sin(2 * np.pi * f1 * n) + np.sin(2 *
np.pi * f2 * n) + np.random.normal(0, 0.1, 1000) *
0.03
samp_freq = 1000
notch_freq = 50
quality_factor = 20
b_notch, a_notch = signal.iirnotch(notch_freq,
quality_factor, samp_freq)
fig, axs = plt.subplots(2, 1, figsize=(8, 6))
axs[0].plot(n, noisysignal, color='red')
axs[0].grid(which='both', axis='both')
axs[0].set_xlabel('Time [s]')
axs[0].set_ylabel('Magnitude')
axs[0].set_title('Noisy Signal')
outputsignal = signal.filtfilt(b_notch, a_notch,
noisysignal)
axs[1].plot(n, outputsignal, color='blue')
axs[1].set_xlabel('Time [s]')
axs[1].set_ylabel('Magnitude')
axs[1].set_title('Filtered Signal')
axs[1].grid()
plt.tight_layout()
plt.show()
freq, h = signal.freqz(b_notch, a_notch, fs=samp_freq)
plt.figure(figsize=(8, 6))
plt.plot(freq, 20 * np.log10(abs(h)), color='blue',
label='Notch Filter')
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude [dB]')
plt.title('Notch Filter Frequency Response')
plt.grid()
plt.legend()
plt.show()
```

12) Design IIR BPF using polezero placement method.
Write python program to plot pole zero
diagram and magnitude response of the same.

code- same as code 5

13) Write python program to design FIR LPF with cutoff
frequency pi/2 and length N=11 using
Rectangular and Blackman window.
Plot the designed filter characteristics.
Verify the filter coefficients (h[n]) by calculation.

Code-

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz, get_window
```

```
# Parameters
N = 11                  # Filter length
cutoff_freq = np.pi / 2  # Cutoff frequency in radians

# Generate the ideal low-pass filter (sinc function)
n = np.arange(0, N)
M = (N - 1) / 2

# Ideal low-pass filter coefficients using sinc
function
h_ideal = np.sinc((n - M) * (cutoff_freq / np.pi))

# Normalize the filter coefficients
h_ideal = h_ideal / np.sum(h_ideal)

# Apply windows
rectangular_window = np.ones(N)
blackman_window = get_window('blackman', N)

# Apply windows to the ideal LPF
h_rectangular = h_ideal * rectangular_window
h_blackman = h_ideal * blackman_window

# Frequency response
w, h_rect = freqz(h_rectangular, 1)
_, h_black = freqz(h_blackman, 1)

# Plot filter coefficients
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.stem(n, h_rectangular, 'b', markerfmt='bo',
basefmt=" ", label="Rectangular Window")
plt.stem(n, h_blackman, 'r', markerfmt='ro', basefmt="
", label="Blackman Window")
plt.title("Filter Coefficients (h[n])")
plt.xlabel("n")
plt.ylabel("h[n]")
plt.legend()
plt.grid()

# Plot frequency response
plt.subplot(2, 1, 2)
plt.plot(w / np.pi, 20 * np.log10(abs(h_rect)), 'b',
label="Rectangular Window")
plt.plot(w / np.pi, 20 * np.log10(abs(h_black)), 'r',
label="Blackman Window")
plt.title("Frequency Response of Designed FIR LPF")
plt.xlabel("Normalized Frequency (×π rad/sample)")
plt.ylabel("Magnitude (dB)")
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

```
# Print the coefficients for verification
print("Ideal LPF Coefficients (h[n]):")
print(h_ideal)
print("\nRectangular Windowed Coefficients
(h_rect[n]):")
print(h_rectangular)
print("\nBlackman Windowed Coefficients
(h_blackman[n]):")
print(h_blackman)
```

14) Design COMB filter using polezero placement method.
Write python program to plot pole zero diagram and magnitude response of the same.

code - same as code number 5

15) Write python program to design FIR HPF with cutoff frequency pi/2 and length N=11 using
Rectangular and Blackman window.
Plot the designed filter characteristics.
Verify the filter coefficients (h[n]) by calculation.

Code-

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz, get_window

# Parameters
N = 11                  # Filter length
cutoff_freq = np.pi / 2  # Cutoff frequency in radians

# Generate the ideal low-pass filter (sinc function)
n = np.arange(0, N)
M = (N - 1) / 2

# Ideal low-pass filter coefficients using sinc
function
h_ideal = np.sinc((n - M) * (cutoff_freq / np.pi))

# Normalize the filter coefficients
h_ideal = h_ideal / np.sum(h_ideal)

# Apply windows
rectangular_window = np.ones(N)
blackman_window = get_window('blackman', N)

# Apply windows to the ideal LPF
```

```
h_rectangular = h_ideal * rectangular_window
h_blackman = h_ideal * blackman_window

# Frequency response
w, h_rect = freqz(h_rectangular, 1)
_, h_black = freqz(h_blackman, 1)

# Plot filter coefficients
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.stem(n, h_rectangular, 'b', markerfmt='bo',
basefmt=" ", label="Rectangular Window")
plt.stem(n, h_blackman, 'r', markerfmt='ro', basefmt="
", label="Blackman Window")
plt.title("Filter Coefficients (h[n])")
plt.xlabel("n")
plt.ylabel("h[n]")
plt.legend()
plt.grid()

# Plot frequency response
plt.subplot(2, 1, 2)
plt.plot(w / np.pi, 20 * np.log10(abs(h_rect)), 'b',
label="Rectangular Window")
plt.plot(w / np.pi, 20 * np.log10(abs(h_black)), 'r',
label="Blackman Window")
plt.title("Frequency Response of Designed FIR LPF")
plt.xlabel("Normalized Frequency (×π rad/sample)")
plt.ylabel("Magnitude (dB)")
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()

# Print the coefficients for verification
print("Ideal LPF Coefficients (h[n]):")
print(h_ideal)
print("\nRectangular Windowed Coefficients
(h_rect[n]):")
print(h_rectangular)
print("\nBlackman Windowed Coefficients
(h_blackman[n]):")
print(h_blackman)
```

16) Write python program to plot window functions in Time Domain and its magnitude spectrum.
Also display window coefficients and verify the same by calculation.

Code-

same as code number 2

17) Design IIR LPF using pole zero placement method. Write python program to plot pole zero diagram and magnitude response of the same.

code - same as code number 5

18) Write python program to plot pole zero diagram of following system. Identify the systems as minimum/maximum/mixed phase.
(A transfer function will be given at the time of exam)

code - same as code number 3

19) Write python program to perform circular convolution of the given two sequences using DFT IDFT method.
x1[n]={ 1,2,3,4} and x2[n]={ 3,5,3,5}
Verify your answers by calculation.
What is the difference between linear and circular convolution?

Code-

```
# Circular convoluθon using DFT
import numpy as np
# Input sequences
x = [1, 2, 3, 4]
h = [1, 2]
# Length of sequences
N1 = len(x)
N2 = len(h)
# Determine the length of the circular convolution
N = max(N1, N2)
# Zero-padding to make lengths equal to N
if N1 < N:
    x = np.pad(x, (0, N - N1), 'constant')
if N2 < N:
    h = np.pad(h, (0, N - N2), 'constant')
print('\n x(n)= ', x)
print('\n h(n)= ', h)
# Compute the DFT of both sequences
XK = np.fft.fft(x)
HK = np.fft.fft(h)
# Circular convolution in the frequency domain
YK = XK * HK
# Compute the inverse DFT to get the circular
convolution result
y = np.fft.ifft(YK)
# Since the output can be complex due to numerical
precision, we take the real part
y = np.real(y)
```

```
print('\n Circular convolution of x(n) and h(n) = ',
y)
```

20) Design IIR HPF using pole zero placement method. Write python program to plot pole zero diagram and magnitude response of the same.

code - same as code number 5

21) Write a python program to synthesize a single sinusoidal signal consisting of frequency components 15 Hz and 50 Hz. Filter this signal using a notch filter having a notch frequency of 50 Hz. Display the original signal, magnitude response of notch filter and filtered signal. (EXP 10)

code - same as code number 11

22) Write python program to plot magnitude spectrum of DFT of 4- point sequence, x[n]={ 2,5,1,3}.
Verify your answers by calculation.
Also plot the magnitude spectrum for N=8, 16 and 32
What is the effect of increase in value of N on DFT spectrum and IDFT

code- same as code number 10