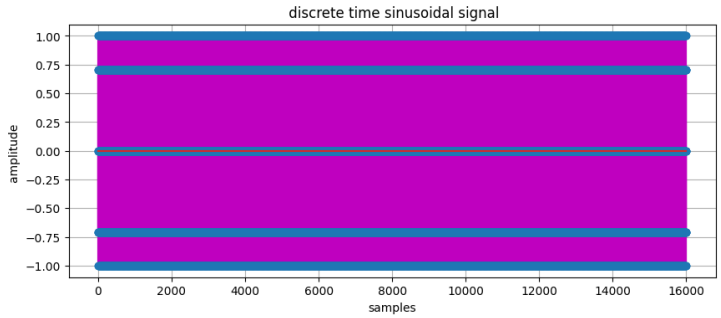
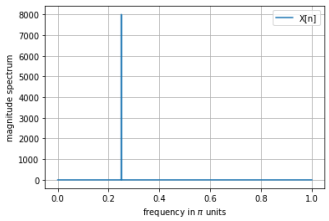
Question.1 (a)

```
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
import math
fs=8000           #sampling frequency
duration=2
num_sample=int(duration*fs)
fm=1/8

n=np.arange(0,num_sample,1)
xn=np.cos((2*math.pi*fm*n))
plt.figure(figsize=(10,4),dpi=100)
plt.stem(n,xn,'m')
plt.title(" discrete time sinusoidal signal")
plt.xlabel("samples")
plt.ylabel("amplitude ")

plt.grid(True)
plt.show()

sf.write('sound8.wav',xn,samplerate=fs)


X=np.abs(np.fft.fft(xn))
X=X[0:len(X)//2]

freq=n*(2*np.pi/len(n))
freq=freq[0:len(freq)//2]

plt.figure()
plt.plot(freq/np.pi,X)
x_mag=np.abs(X)
plt.legend(['X[n]'])
plt.xlabel('frequency in $\pi$ units')
plt.ylabel('magnitude spectrum')
plt.grid('True')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of inc
```

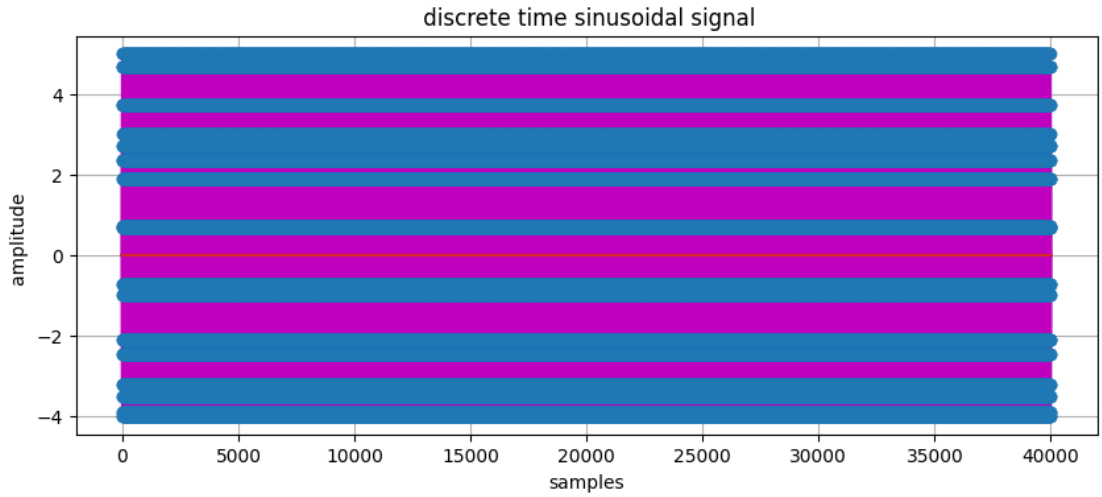


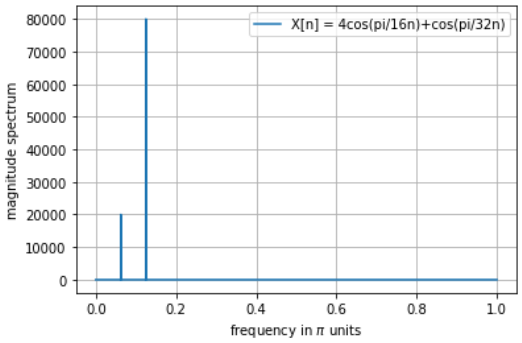


Question.2

```
import soundfile as sf
#import pysoundfile as psf
import numpy as np
import matplotlib.pyplot as plt
import math
fs=8000             #sampling frequency
duration=5          #in sec
num_sample=int(duration*fs)

fm1=1/16
n=np.arange(0,num_sample,1)
xn1=4*np.cos((2*math.pi*fm1*n))

fm2=1/32
n=np.arange(0,num_sample,1)
xn2=np.cos((2*math.pi*fm2*n))

xn=xn1+xn2

plt.figure(figsize=(10,4),dpi=100)
plt.stem(n,xn,'m')
plt.title(" discrete time sinusoidal signal")
```

```
plt.xlabel("samples")
plt.ylabel("amplitude ")

plt.grid(True)
plt.show()

sf.write('sound2_1.wav',xn,samplerate=fs)


X=np.abs(np.fft.fft(xn))
X=X[0:len(X)//2]

freq=n*(2*np.pi/len(n))
freq=freq[0:len(freq)//2]

plt.figure()
plt.plot(freq/np.pi,X)
x_mag=np.abs(X)
plt.legend(['X[n] = 4cos(pi/16n)+cos(pi/32n)'])
plt.xlabel('frequency in $\pi$ units')
plt.ylabel('magnitude spectrum')
plt.grid('True')
plt.show()
```





Question. 3

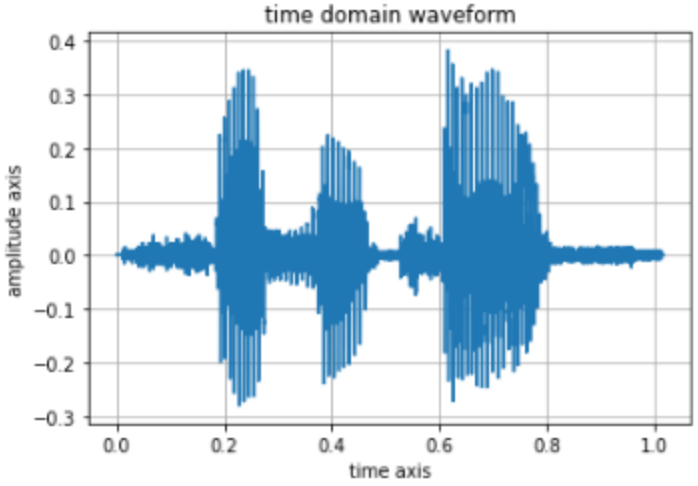
```
from scipy import signal
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt

def enframe(x,winsize,hoplength,fs,wintype):
    frame_length = winsize * fs    #n
    frame_step = hoplength * fs    #r
    signal_length = len(x)
    frames_overlap = frame_length - frame_step    #n-r
    num_frames = np.abs(signal_length - frames_overlap) //
np.abs(frame_length - frames_overlap)
    rest_samples = np.abs(signal_length - frames_overlap) %
np.abs(frame_length - frames_overlap)
    if rest_samples != 0:
        pad_signal_length = int(frame_step - rest_samples)
        z = np.zeros((pad_signal_length))
        pad_signal = np.append(x, z)
```

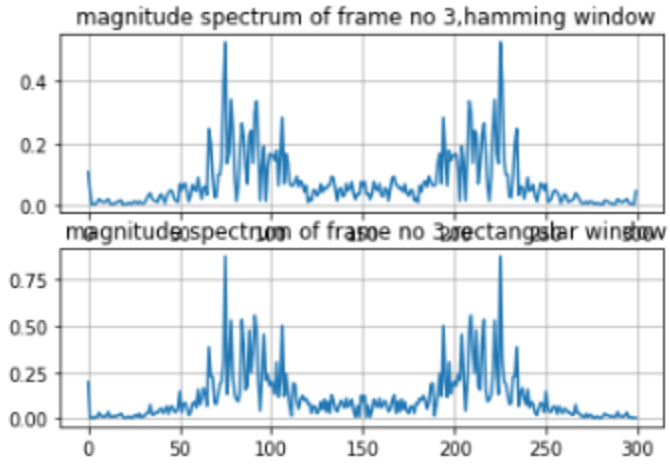
```
        num_frames += 1
    else:
        pad_signal = x
    frame_length = int(frame_length)
    frame_step = int(frame_step)
    frame_no= int(num_frames)

    idx1 = np.tile(np.arange(0, frame_length), (frame_no, 1))
    idx2 = np.tile(np.arange(0, num_frames * frame_step, frame_step),
                   (frame_length, 1)).T
    indices = idx1 + idx2
    frames = pad_signal[indices.astype(np.int32, copy=False)]
    if wintype=="hamm":
        window = signal.windows.hamming(frame_length)
    elif wintype=="rect":
        window=np.ones(frame_length)
    frame_window=frames*window
    return frame_window


data,sr1=sf.read('should.wav')
print("sampling rate",sr1)


# plotting time domain waveform
t=np.linspace(0,len(data)/sr1,len(data))
plt.figure()
plt.plot(t,data),plt.xlabel("time axis"),plt.ylabel("amplitude
axis"),plt.title("time domain waveform"),plt.grid("True")
#framing and windowing
#for hamming window
frame_window1=enframe(data,0.03,0.015,sr1,"hamm")
print("hamming window",frame_window1)
n=len(frame_window1)
x1=[]
y1=[]
x2=[]
y2=[]
for i in range(n):
   x1.append(np.abs(np.fft.fft(frame_window1[i])))
```

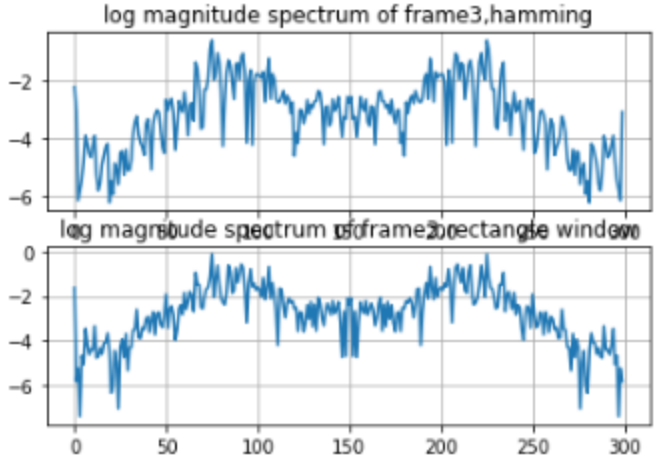
```
    y1.append(np.log(np.abs(np.fft.fft(frame_window1[i]))))
#rectangular window
frame_window2=enframe(data,0.03,0.015,sr1,"rect")
print("rectangular window",frame_window2)

n=len(frame_window2)
for i in range(n):
    x2.append(np.abs(np.fft.fft(frame_window2[i])))
    y2.append(np.log(np.abs(np.fft.fft(frame_window2[i]))))


i=int(input("enter any frame number to see it's frequency spectrum : "))

plt.figure(),
plt.subplot(2,1,1),plt.plot(x1[i])  ,plt.grid("True"),plt.title("
magnitude spectrum of frame no {},hamming
window".format(i)),plt.grid("True")

plt.subplot(2,1,2),plt.plot(x2[i]),plt.grid('True') ,plt.title(" magnitude
spectrum of frame no {},rectangular window".format(i)),plt.grid("True")
plt.figure(),
plt.subplot(2,1,1),plt.plot(y1[i]) ,plt.grid('True') ,plt.title("log
magnitude spectrum of frame{},hamming".format(i))
plt.subplot(2,1,2),plt.plot(y2[i])
plt.grid('True')
plt.title("log magnitude spectrum of frame{},rectangle window".format(i))
plt.grid("True")
```

```
sampling rate 10000
hamming window [[ 3.17382813e-05  3.66675849e-05  9.81521011e-06 ...
1.69312374e-04
  -5.13346189e-04 -1.78222656e-04]
 [-3.75976563e-04 -1.83337925e-04  5.76643594e-04 ... -1.34959139e-04
   2.10227487e-04  2.75878906e-04]
 [-4.93164063e-04 -2.10227487e-04 -6.37988657e-05 ... -9.69251999e-04
   1.82360122e-03 -5.68847656e-04]
 ...
 [-2.53906250e-04  4.37566513e-04  1.61950967e-04 ...  3.16540526e-04
   3.42230793e-04 -4.80957031e-04]
 [ 5.02929688e-04 -6.16015427e-04  5.61920779e-04 ...  1.42320547e-04
```

```
   1.24669789e-04  1.04980469e-04]
 [-1.12304688e-04  5.32902234e-04  3.92608405e-05 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
rectangular window [[ 0.00039673  0.00045776  0.00012207 ...  0.00210571
-0.00640869
  -0.00222778]
 [-0.00469971 -0.00228882  0.00717163 ... -0.00167847  0.00262451
   0.00344849]
 [-0.00616455 -0.00262451 -0.00079346 ... -0.01205444  0.02276611
  -0.0071106 ]
 ...
 [-0.00317383  0.00546265  0.00201416 ...  0.00393677  0.00427246
  -0.00601196]
 [ 0.00628662 -0.00769043  0.00698853 ...  0.00177002  0.0015564
   0.00131226]
 [-0.00140381  0.00665283  0.00048828 ...  0.          0.
   0.        ]]
enter any frame number to see it's frequency spectrum : 3
```







Question. 4

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

```
def enframe(x,winsize,hoplength,fs,wintype):
    frame_length = winsize * fs
    frame_step = hoplength * fs
    signal_length = len(x)
    frames_overlap = frame_length - frame_step
    num_frames = np.abs(signal_length - frames_overlap) //
np.abs(frame_length - frames_overlap)
    rest_samples = np.abs(signal_length - frames_overlap) %
np.abs(frame_length - frames_overlap)
    if rest_samples != 0:
        pad_signal_length = int(frame_step - rest_samples)
        z = np.zeros((pad_signal_length))
        pad_signal = np.append(x, z)
        num_frames += 1
    else:
        pad_signal = x
    frame_length = int(frame_length)
    frame_step = int(frame_step)
    frame_no= int(num_frames)

    idx1 = np.tile(np.arange(0, frame_length), (frame_no, 1))
    idx2 = np.tile(np.arange(0, num_frames * frame_step, frame_step),
                   (frame_length, 1)).T
    indices = idx1 + idx2
    frames = pad_signal[indices.astype(np.int32, copy=False)]
    if wintype=="hamm":
        window = signal.windows.hamming(frame_length)
    elif wintype=="rect":
        window=np.ones(frame_length)
    frame_window=frames*window
    return frame_window
def shortTermEnergy(frame_window):
    energy=[]
    avg1=[]
    for x in frame_window:
        y=sum(abs(x)**2)
        energy.append(y)
        avg1.append(y/len(x))
```

```
    return energy
data,sr1=sf.read('should.wav')
frame_window=enframe(data,0.03,0.015,sr1,"rect")
g=shortTermEnergy(frame_window)
plt.plot(g),plt.grid("True"),plt.title("energy
spectrum"),plt.xlabel("frequency"),plt.ylabel("energy")
Sum=0
n=len(frame_window)
for en in g:
    Sum+=en/300;
avg=Sum
print("average is:",avg)
x=[]

start=[]
end=[]
for i in range(len(g)-1):
    if g[i]<avg and g[i+1]>avg:
        start.append(i)
    if g[i]>avg and g[i+1]<avg:
        end.append(i)
segment1=frame_window[start[0]:end[0]]
segment_1=[]
for i in segment1:
    for j in range(len(i)):
        segment_1.append(i[j])
sf.write("should_segmentation_1.wav",segment_1,samplerate=sr1)
segment2=frame_window[start[1]:end[1]]
segment_2=[]
for i in segment2:
    for j in range(len(i)):
        segment_2.append(i[j])
sf.write("should_segmentation_2.wav",segment_2,samplerate=sr1)
segment3=frame_window[start[2]:end[2]]
segment_3=[]
for i in segment3:
    for j in range(len(i)):
        segment_3.append(i[j])
sf.write("should_segmentation_3.wav",segment_3,samplerate=sr1)
```

```
average is: 0.2737285591444622
```