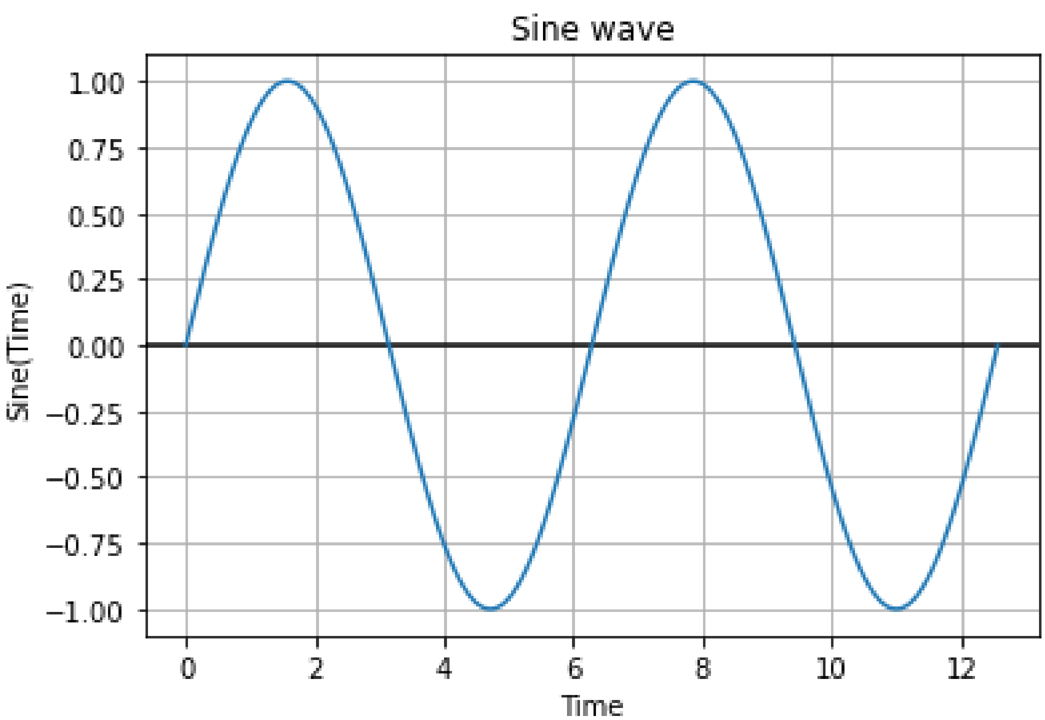
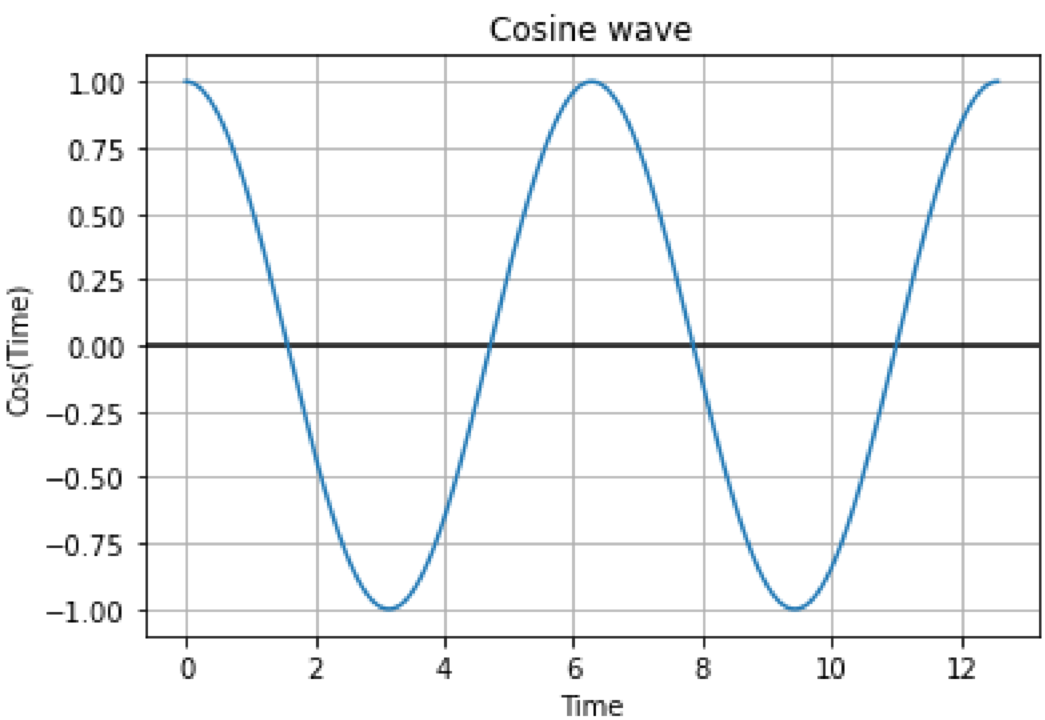
```
#Q1 (a)

import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x = np.arange(0,4*np.pi,0.001)    #this line makes an array from 0 to
4pi, with a step of 0.001
y = np.sin(x)                     #this calculates sin of all values
in our array

plt.title('Sine wave')            #this line gives the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Sine(Time)')          #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph

plt.show()                        #this line shows the graph and
successfully terminates the program
```



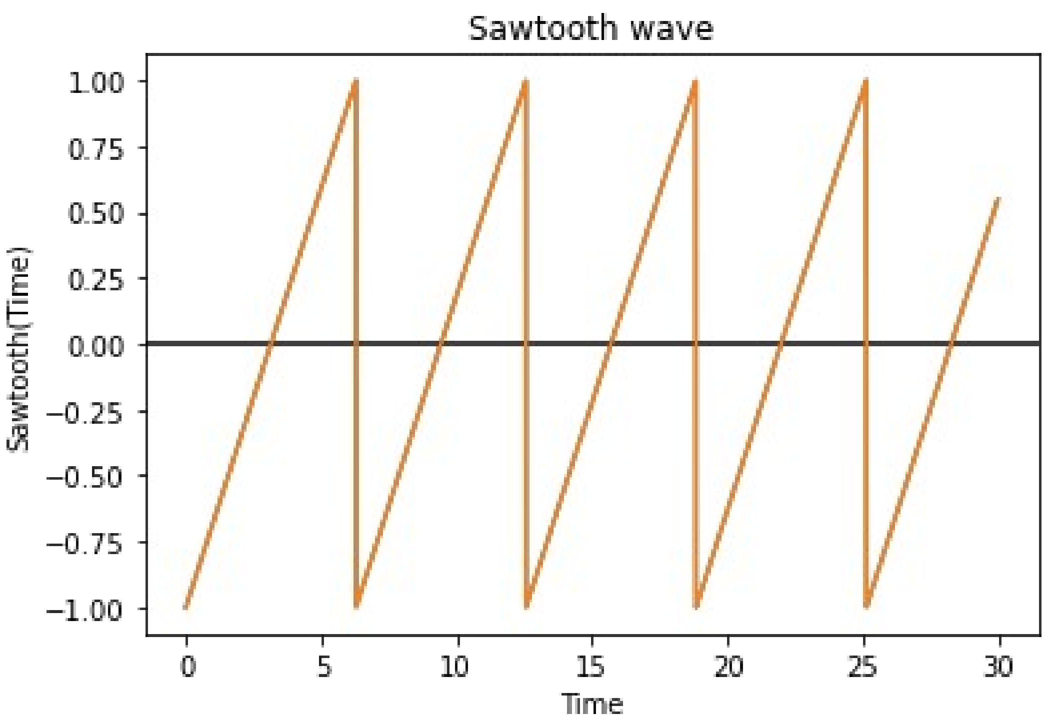

```
#Q1 (b)
```

```
import matplotlib.pyplot as plt   #importing matplotlib library
import numpy as np                #importing numpy library

x = np.arange(0,4*np.pi,0.001)    #this line makes an array from 0 to
4pi, with a step of 0.001
y = np.cos(x)                     #this calculates cos of all values
in our array

plt.title('Cosine wave')          #this line gives the plot a title
plt.xlabel('Time')                #this line gives a name for the x
axis and its variables
plt.ylabel('Cos(Time)')           #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')      #this line generates a horizontal
line at y=0, with color code as black

plt.grid()                        #this line generates gridlines,
which adds more detail to the graph
plt.plot(x,y)                     #this line plots the graph

plt.show()                        #this line shows the graph and
successfully terminates the program
```



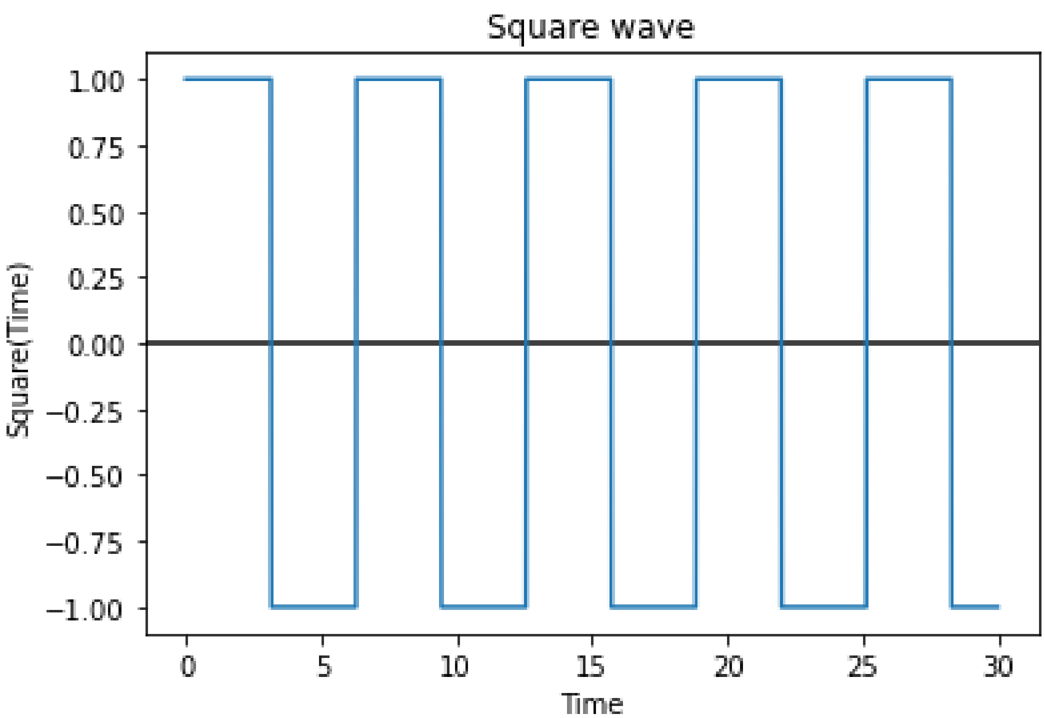

```
#Q1 (c)

import matplotlib.pyplot as plt    #importing matplotlib library
from scipy import signal           #importing signal library from
```

```
scipy
import numpy as np                      #importing numpy library

x = np.arange(0,30,0.01)                #this line makes an array from 0
to 30, with a step of 0.01
y = signal.sawtooth(x)                  #this line generates the sawtooth
waveform for the values in our array

plt.title('Sawtooth wave')              #this line gives the plot a title

plt.xlabel('Time')                      #this line gives a name for the x
axis and its variables
plt.ylabel('Sawtooth(Time)')            #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')            #this line generates a horizontal
line at y=0, with color code as black

plt.plot(x,y)                           #this line plots the graph
plt.show()                              #this line shows the graph and
successfully terminates the program
```



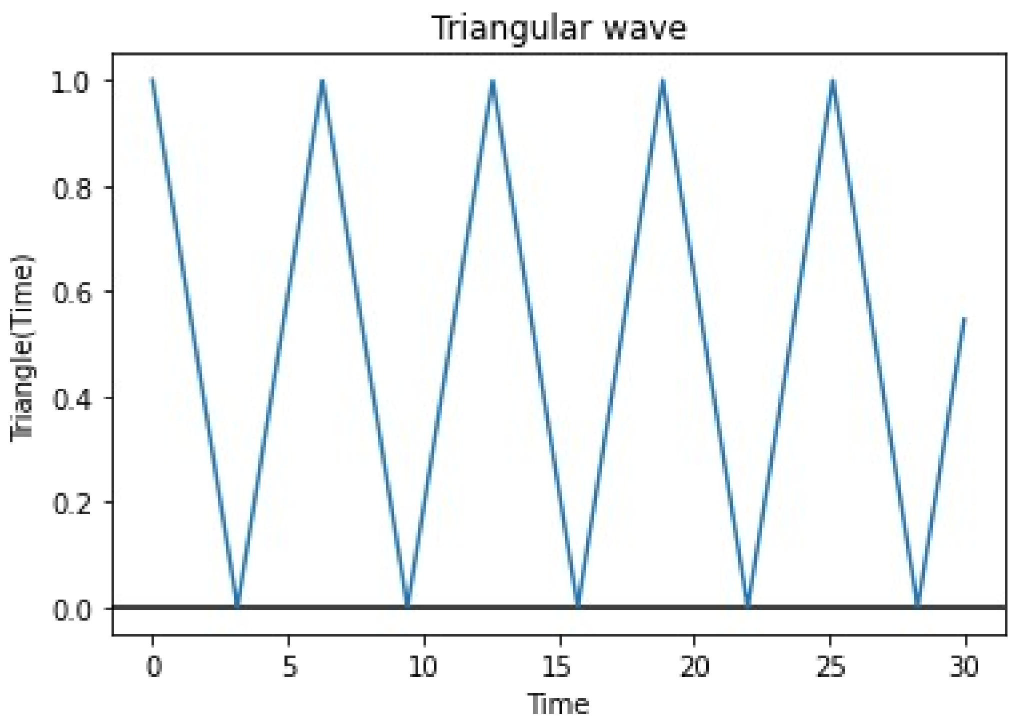

```
#Q1 (d)

import matplotlib.pyplot as plt         #importing matplotlib library
from scipy import signal                #importing signal library from
scipy
import numpy as np                      #importing numpy library
```

```
x = np.arange(0,30,0.01)            #this line makes an array from 0
to 30, with a step of 0.01
y = signal.square(x)                #this line generates the square
waveform for the values in our array

plt.title('Square wave')            #this line gives the plot a title

plt.xlabel('Time')                  #this line gives a name for the x
axis and its variables
plt.ylabel('Square(Time)')          #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')        #this line generates a horizontal
line at y=0, with color code as black

plt.plot(x,y)                       #this line plots the graph
plt.show()                          #this line shows the graph and
successfully terminates the program
```



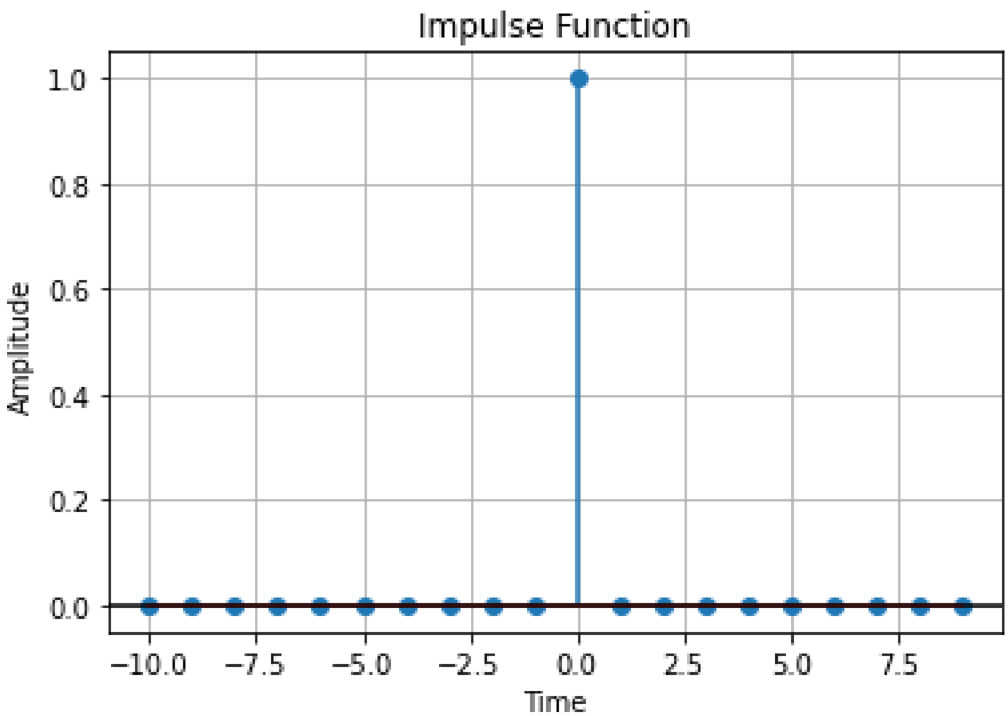

```
#Q1 (e)

import matplotlib.pyplot as plt     #importing matplotlib library
from scipy import signal            #importing signal library from
scipy
import numpy as np                  #importing numpy library

x = np.arange(0,30,0.01)            #this line makes an array from 0
```

```
to 30, with a step of 0.01
y = abs(signal.sawtooth(x))         #this line generates the
triangular waveform for the values in our array by applying abs
function

plt.title('Triangular wave')        #this line gives the plot a title

plt.xlabel('Time')                  #this line gives a name for the x
axis and its variables
plt.ylabel('Triangle(Time)')        #this line gives a name for the y
axis and its variables
plt.axhline(y=0,color = 'k')        #this line generates a horizontal
line at y=0, with color code as black

plt.plot(x,y)                       #this line plots the graph
plt.show()                          #this line shows the graph and
successfully terminates the program
```



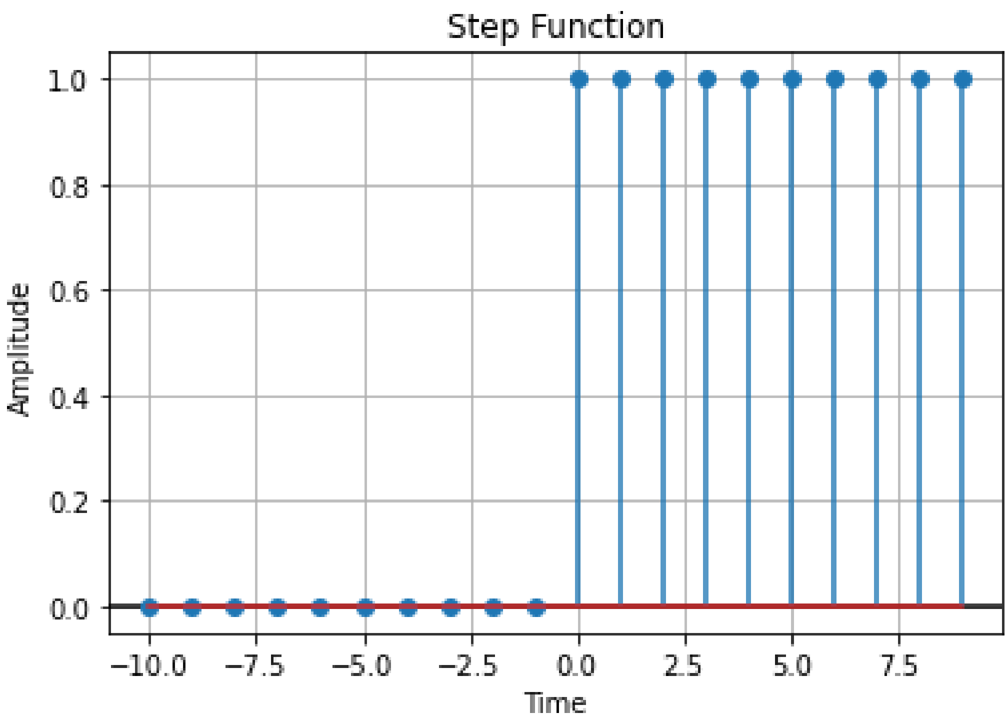

```
#Q2 (a)

import matplotlib.pyplot as plt            #importing matplotlib
library
import numpy as np                         #importing numpy library

x=np.arange(-10,10)                        #this line makes an array
from -10 to 10, with a step of 1
y=[]                                       #this line creates an empty
```

```
list
for i in x:                                     #for loop to traverse the
list x
  if i==0:                                      #if condition to check if
element is 0 in x
    y.append(1)                                 #append 1 in list y
  else:                                         #else condition
    y.append(0)                                 #append 0 in list y

plt.axhline(y=0, color="black")
plt.title("Impulse Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.stem(x,y, use_line_collection=True)    #this line is the stem
function, it is used to plot discrete time signals
plt.grid()
plt.show()
```



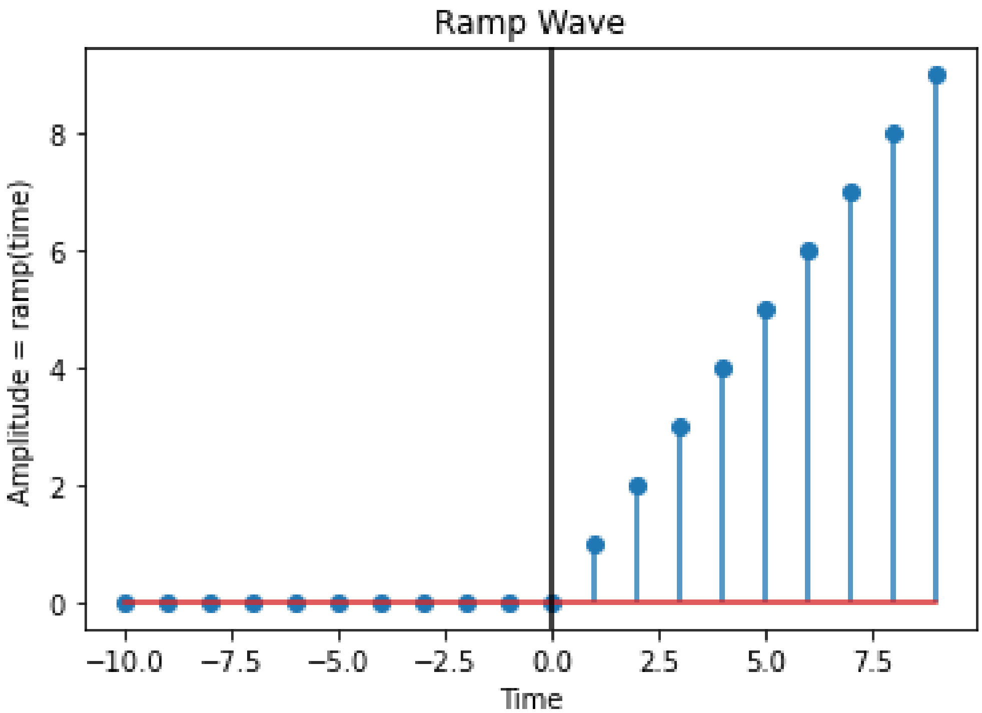

```
#Q2 (b)

import matplotlib.pyplot as plt   #importing matplolib library
import numpy as np

x=np.arange(-10,10)
y=[]                              #logic to get the values of
dependant variables for plotting Step Signal
for i in x:
  if (i<0):
```

```
    y.append(0)
  else:
    y.append(1)

plt.axhline(y=0, color="black")
plt.title("Step Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")

plt.grid()
plt.stem(x,y,use_line_collection="True")
plt.show()
```

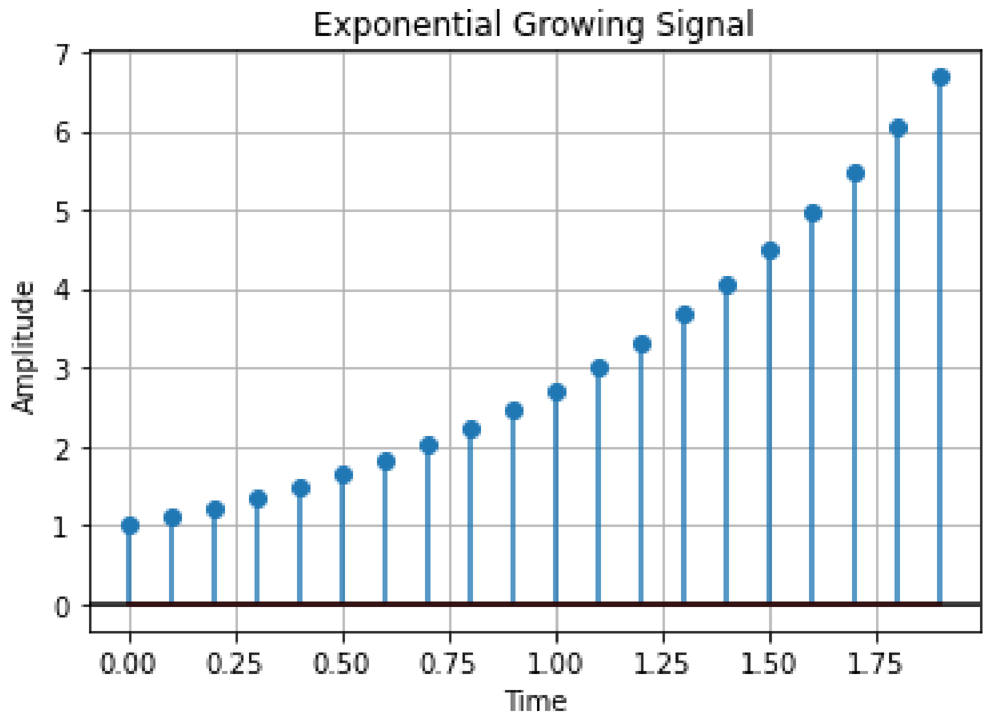



```
#Q2 (c)

import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-10,10)
y=[]
for i in x:                                #logic to get the values of 
dependant variables for plotting Ramp signal
  if (i<=0):
    y.append(0)
  elif (i>0):
    y.append(i)
```

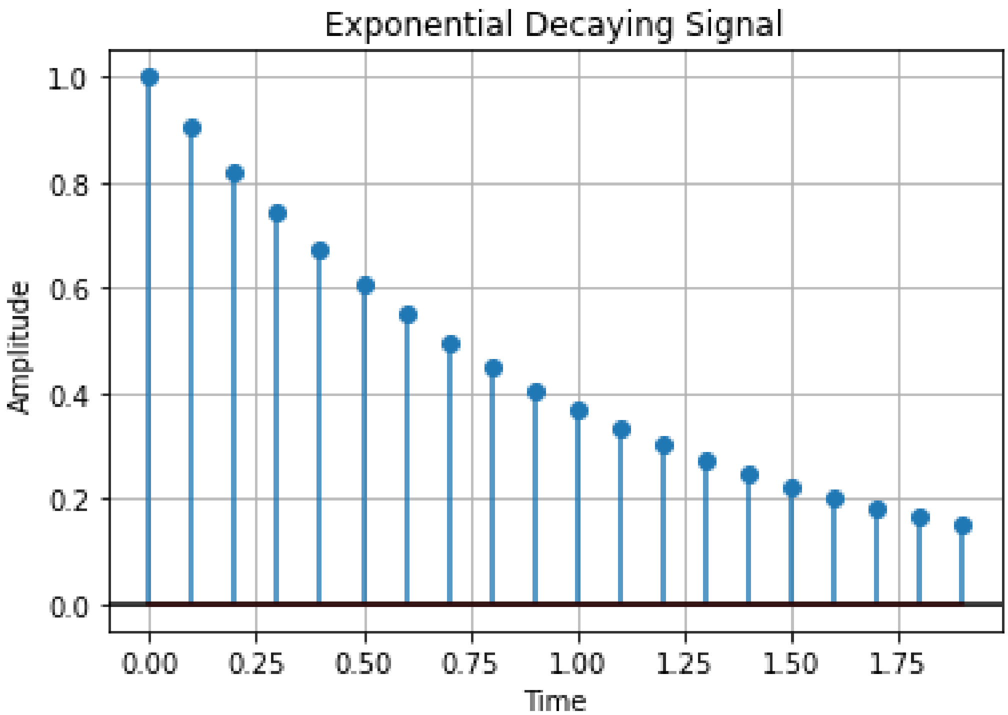
```
plt.stem(x,y,use_line_collection="True")
plt.grid()
plt.axhline(y=0, color="black")
plt.title("Ramp Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.show()
```



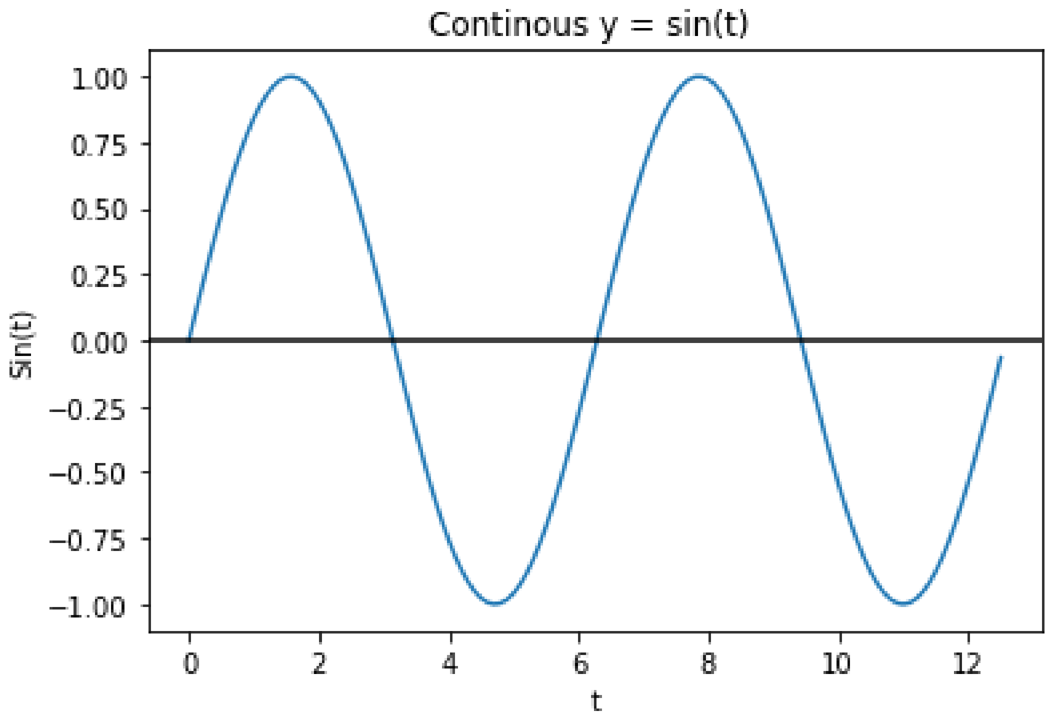

```
#Q2 (d)

import matplotlib.pyplot as plt
import numpy as np
time = np.arange(0,2,0.1);
amp = np.exp(time)                                  #inbuilt exp function
uses from numpy
plt.stem(time,amp,use_line_collection='true')
plt.title('Exponential Growing Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.axhline(y=0,color='k')
plt.grid()
plt.show()
```

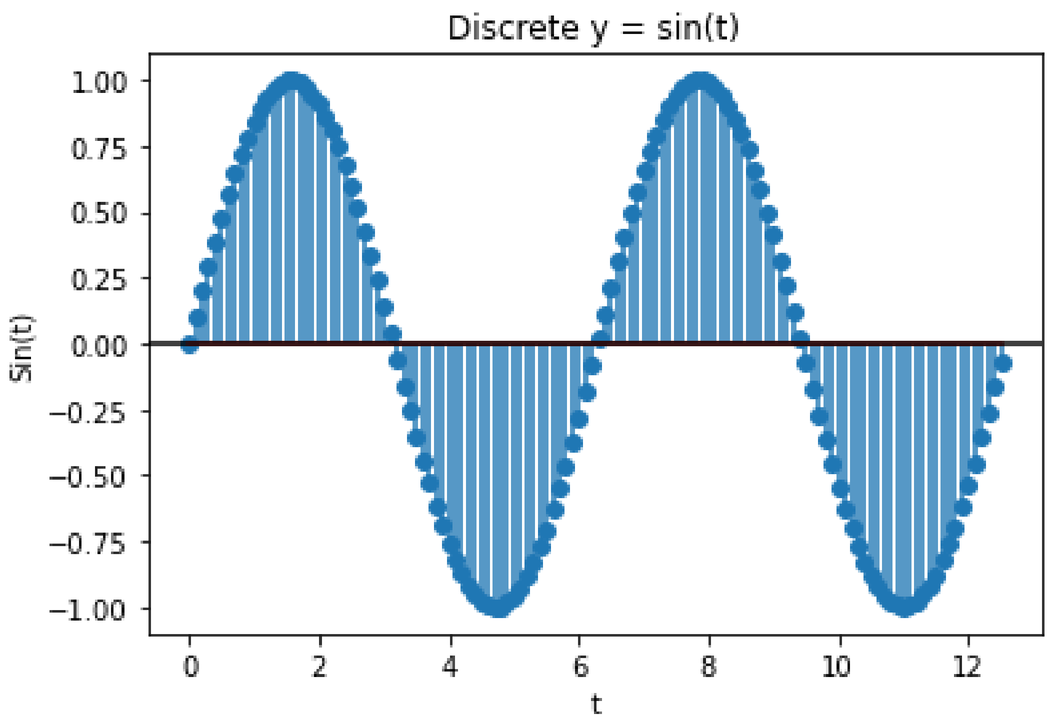



```
#Q2 (e)

import matplotlib.pyplot as plt
import numpy as np
time = np.arange(0,2,0.1);
amp = np.exp(-time)                                #inbuilt exp
function from numpy
plt.stem(time,amp,use_line_collection='true')
plt.title('Exponential Decaying Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.axhline(y=0,color='k')
plt.grid()
plt.show()
```



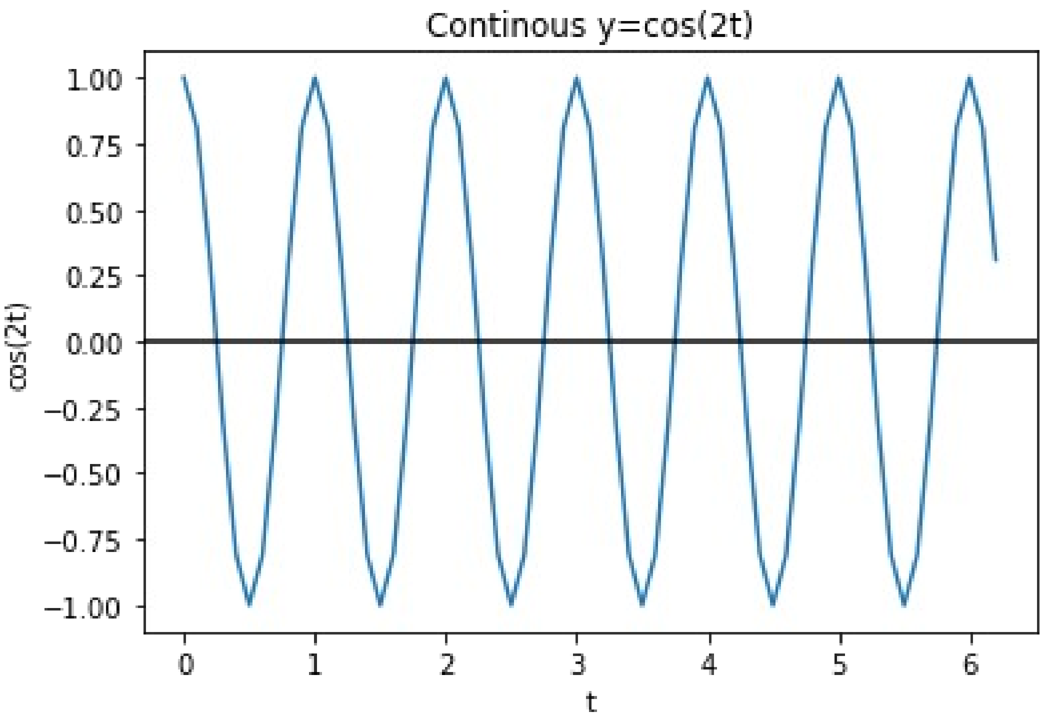

```
#Q3 (a)


import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,4*np.pi,0.1)
y = np.sin(x)
plt.plot(x,y)
plt.title('Continous y = sin(t) ')
plt.xlabel('t')
plt.ylabel('Sin(t)')
plt.axhline(y=0,color='k')
plt.show()
plt.stem(x,y,use_line_collection='true' )
plt.title('Discrete y = sin(t)')
plt.xlabel('t')
plt.ylabel('Sin(t)')
plt.axhline(y=0,color='k')
plt.show()
```

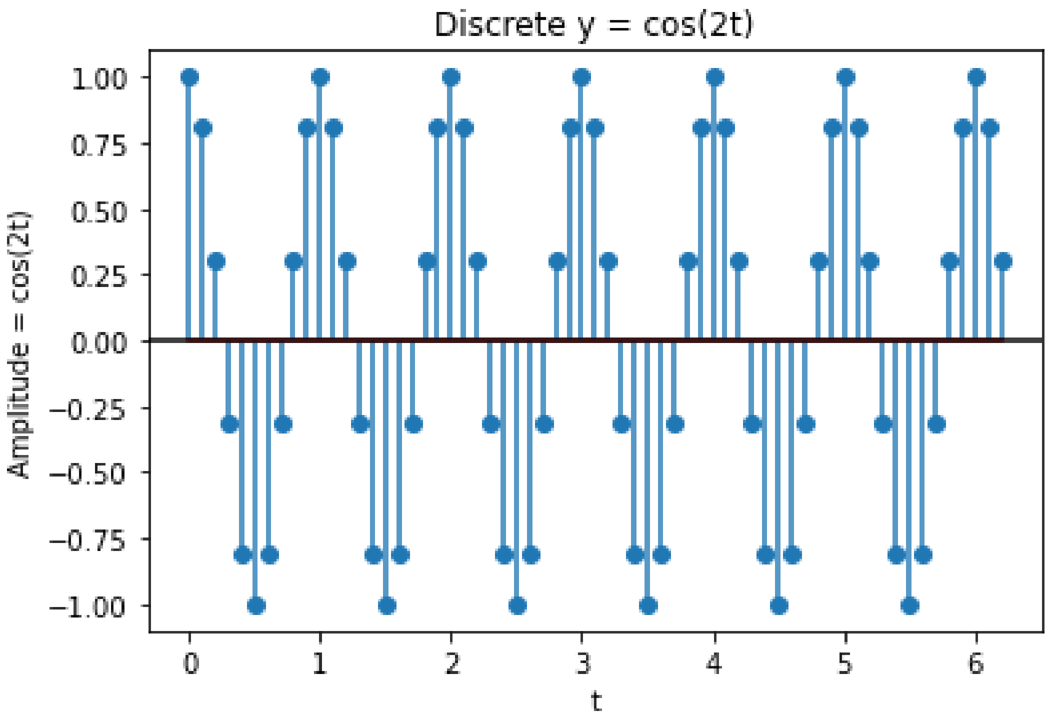




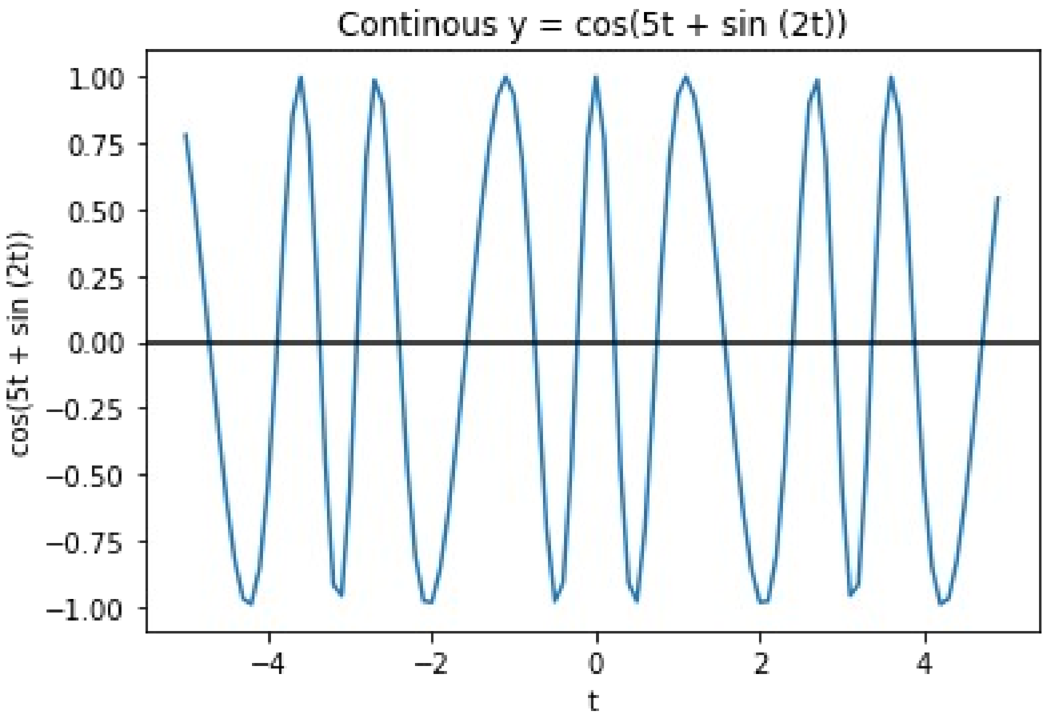

```
#Q3 (b)

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

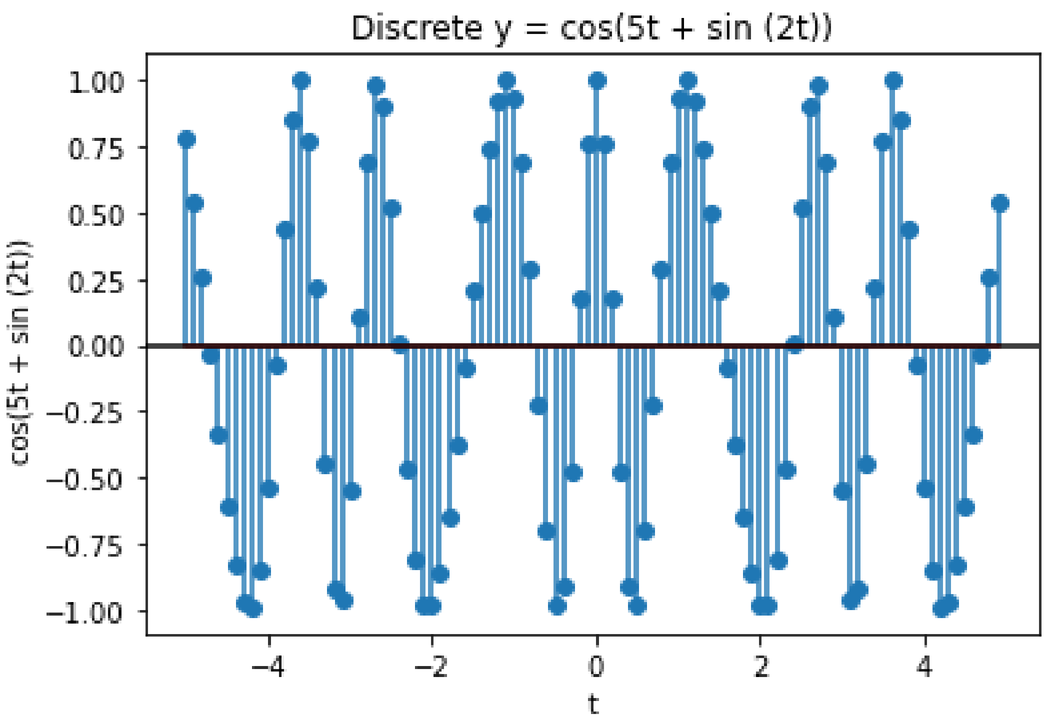
```
x = np.arange(0,2*np.pi,0.1)
y = np.cos(2*np.pi*x)
plt.plot(x,y)
plt.title('Continous y=cos(2t)')
plt.xlabel('t')
plt.ylabel('cos(2t)')
plt.axhline(y=0,color='k')
plt.show()
plt.stem(x,y,use_line_collection='true' )
plt.title('Discrete y = cos(2t)')
plt.xlabel('t')
plt.ylabel('Amplitude = cos(2t)')
plt.axhline(y=0,color='k')
plt.show()
```





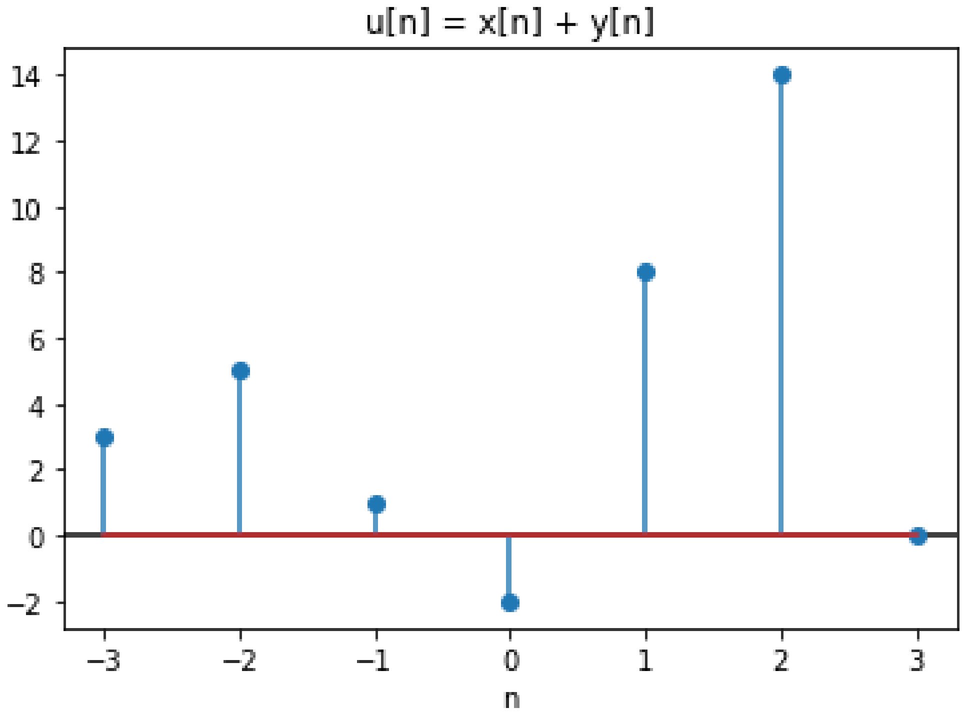

```
#Q3 (c)

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

x = np.arange(-5,5,0.1)
y = np.cos(np.sin(2*time) + (5 * time))
plt.plot(x,y)
plt.title('Continous y = cos(5t + sin (2t))')
plt.xlabel('t')
plt.ylabel('cos(5t + sin (2t))')
plt.axhline(y=0,color='k')
plt.show()
plt.stem(time,amp,use_line_collection='true' )
plt.title('Discrete y = cos(5t + sin (2t))')
plt.xlabel('t')
plt.ylabel('cos(5t + sin (2t)) ')
plt.axhline(y=0,color='k')
plt.show()
```





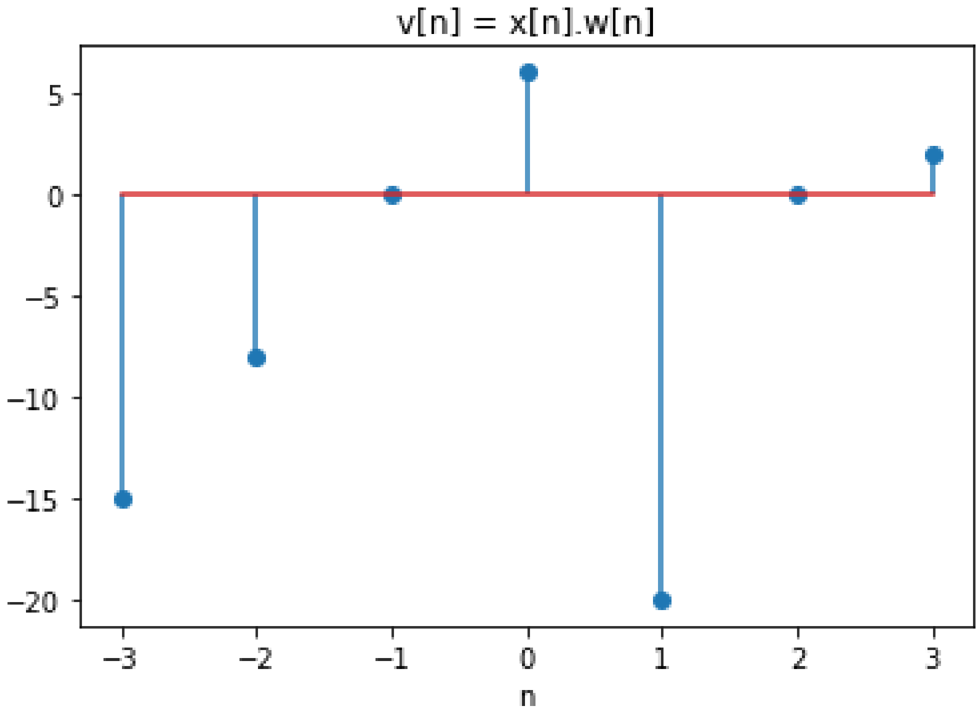

```
#Q4 (a)

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

```
row = np.arange(-3,4,1)                         #this command gets
discrete values between -3 and 3, with a step of 1
x = np.array([3,-2,0,1,4,5,2])                  #initialising the 3
arrays given in the question
y = np.array([0,7,1,-3,4,9,-2])
w = np.array([-5,4,3,6,-5,0,1])

sumarr = x+y                                    #resultant array is the
sum of the arrays x and y

plt.axhline(y=0,color='k')
plt.title('u[n] = x[n] + y[n]')
plt.xlabel('n')
plt.stem(row,sumarr)
plt.show()
```



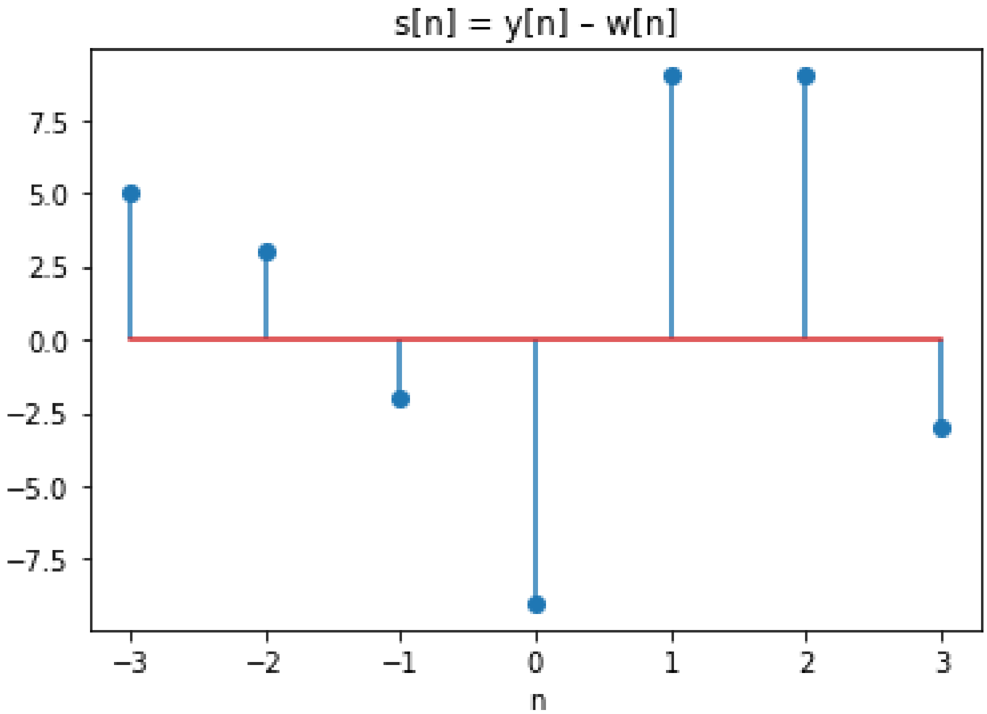

```
#Q4 (b)

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

row = np.arange(-3,4,1)
x = np.array([3,-2,0,1,4,5,2])
y = np.array([0,7,1,-3,4,9,-2])
w = np.array([-5,4,3,6,-5,0,1])
```

```
multarr = x*w                              #resultant array is the
product of the arrays x and w

plt.axhline(y=0,color='k')
plt.title('v[n] = x[n].w[n]')
plt.xlabel('n')
plt.stem(row,multarr)
plt.show()
```

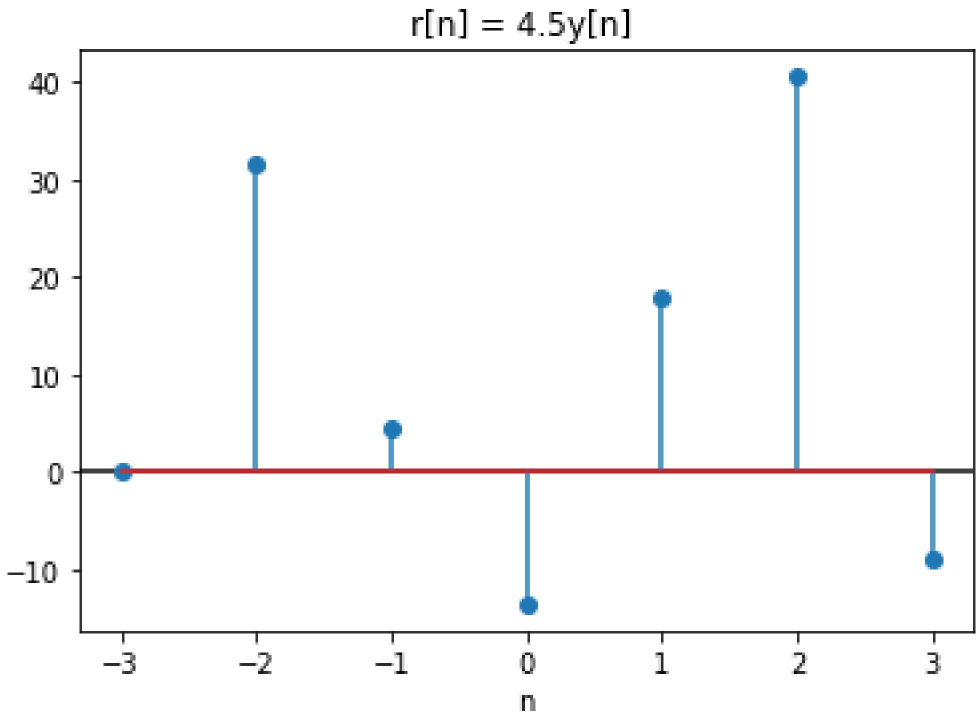



```
#Q4 (c)

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

row = np.arange(-3,4,1)
x = np.array([3,-2,0,1,4,5,2])
y = np.array([0,7,1,-3,4,9,-2])
w = np.array([-5,4,3,6,-5,0,1])

diffarr = y-w                               #resultant array is the
difference of the arrays y and w

plt.axhline(y=0,color='k')
plt.title('s[n] = y[n] – w[n]')
plt.xlabel('n')
```

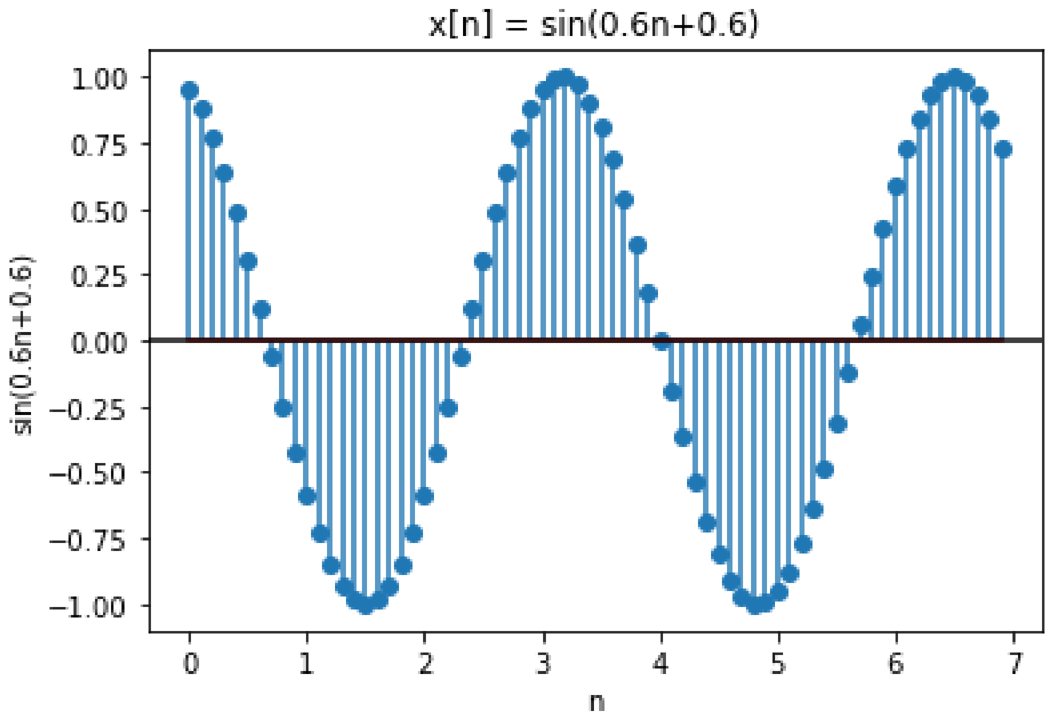
```
plt.stem(row,diffarr)
plt.show()
```



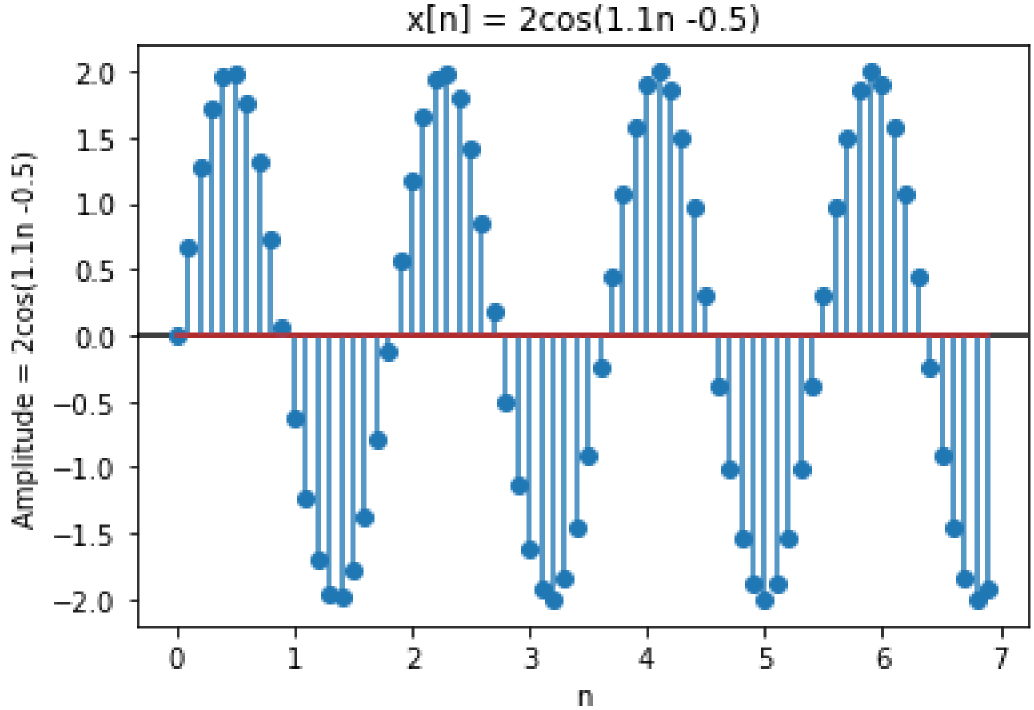

```
#Q4 (d)

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

row = np.arange(-3,4,1)
x = np.array([3,-2,0,1,4,5,2])
y = np.array([0,7,1,-3,4,9,-2])
w = np.array([-5,4,3,6,-5,0,1])

newarr = 4.5*y                        #resultant array is the product
of the scalar 4.5 and y

plt.axhline(y=0,color='k')
plt.title('r[n] = 4.5y[n] ')
plt.xlabel('n')
plt.stem(row,newarr)
plt.show()
```



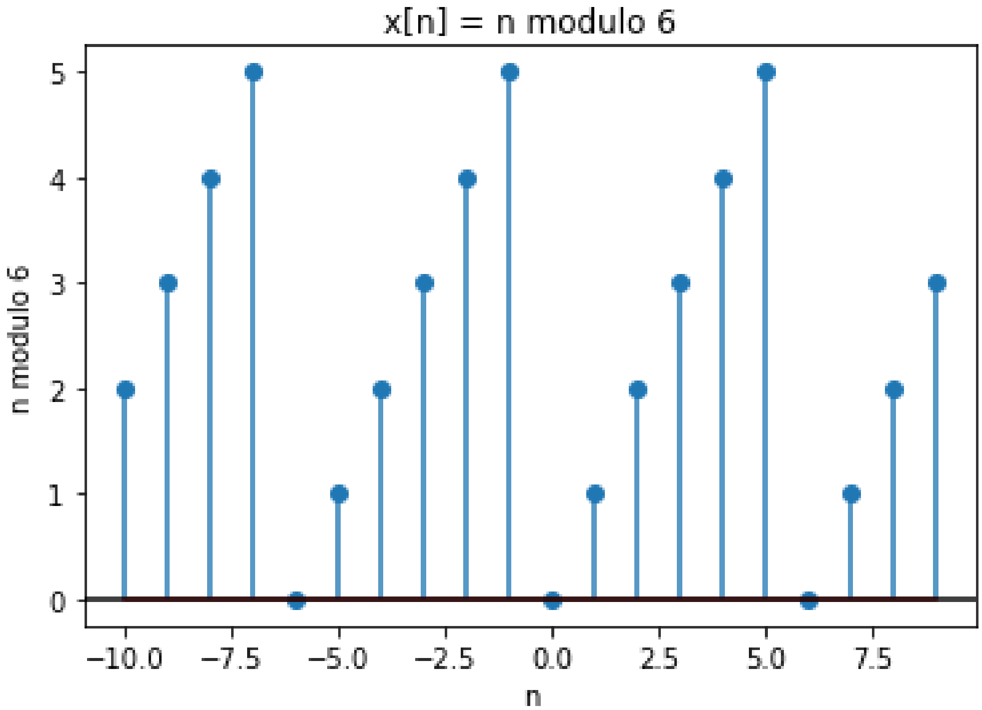

```
#Q5 (a)

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,7,0.1)

y = np.sin(0.6*np.pi*x + 0.6*np.pi)          #function to generate
the given signal

plt.title('x[n] = sin(0.6n+0.6)')

plt.xlabel('n')
plt.ylabel('sin(0.6n+0.6)')

plt.stem(x,y)
plt.axhline(y=0,color='k')
plt.show()
```




```
#Q5 (b)

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,7,0.1)

y = 2*np.cos(1.1*np.pi*x - 0.5*np.pi)        #function to generate the 
given signal

plt.title('x[n] = 2cos(1.1n -0.5)')

plt.xlabel('n')
plt.ylabel('Amplitude = 2cos(1.1n -0.5)')

plt.axhline(y=0,color='k')
plt.stem(x,y)
plt.show()
```




```
#Q5 (c)

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

x = np.arange(-10,10,1);
y = np.mod(x,6)                                   #function to generate the
given signal
plt.stem(x,y)
plt.title('x[n] = n modulo 6')
plt.xlabel('n')
plt.ylabel('n modulo 6')
plt.axhline(y=0,color='k')
plt.show()
```



x[n] = n modulo 6
n modulo 6
n
0
1
2
3
4
5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5