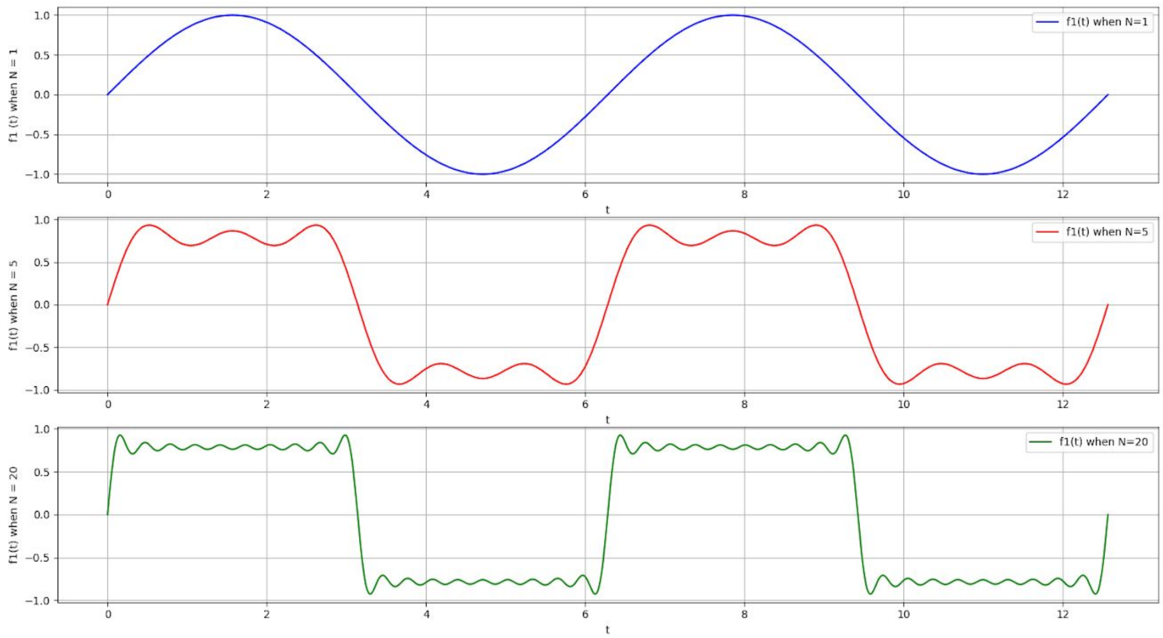
Solution for Q.1:

```
# Q.1_(a).py  (Part (a) in Question.1) :

import numpy as np
import matplotlib.pyplot as plt
from pylab import *
t = np.linspace(0,4*np.pi,1000)

def Fourier_cal(N):
	y = 0
	for n in range(1,N+1,2):
		y = y + (np.sin(n*t))/(n)
	return y
y1 = Fourier_cal(1)
y2 = Fourier_cal(5)
y3 = Fourier_cal(20)

subplot(3,1,1)
plt.plot(t,y1,'b',label='f1(t) when N=1')
plt.grid()
plt.xlabel('t')
plt.ylabel('f1 (t) when N = 1')
plt.legend()

subplot(3,1,2)
plt.plot(t,y2,'r',label='f1(t) when N=5')
plt.grid()
plt.xlabel('t')
plt.ylabel('f1(t) when N = 5')
plt.legend()

subplot(3,1,3)
plt.plot(t,y3,'g',label='f1(t) when N=20')
plt.grid()
plt.xlabel('t')
plt.ylabel('f1(t) when N = 20')
plt.legend()
plt.show()
```

Figure Obtained after running the code:

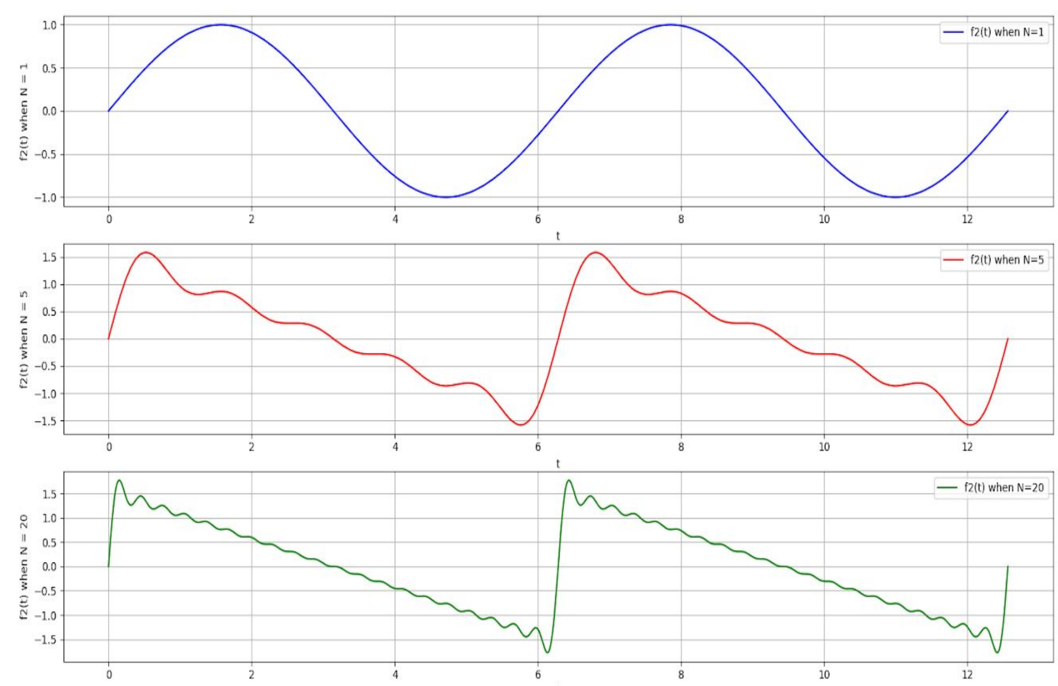


Q.1_(b).py (Part (b) in Question.1) :

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
t = np.linspace(0,4*np.pi,1000)
#print(t)
def Fourier_cal(N):
        y = 0
        for n in range(1,N+1,1):
                y = y + (np.sin(n*t))/(n)
        return y
y1 = Fourier_cal(1)
y2 = Fourier_cal(5)
y3 = Fourier_cal(20)

subplot(3,1,1)
plt.plot(t,y1,'b',label='f2(t) when N=1')
plt.grid()
plt.xlabel('t')
```

```
plt.ylabel('f2(t) when N = 1')
plt.legend()

subplot(3,1,2)
plt.plot(t,y2,'r',label='f2(t) when N=5')
plt.grid()
plt.xlabel('t')
plt.ylabel('f2(t) when N = 5')
plt.legend()

subplot(3,1,3)
plt.plot(t,y3,'g',label='f2(t) when N=20')
plt.grid()
plt.xlabel('t')
plt.ylabel('f2(t) when N = 20')
plt.legend()
plt.show()
```

Figure Obtained after running the code:

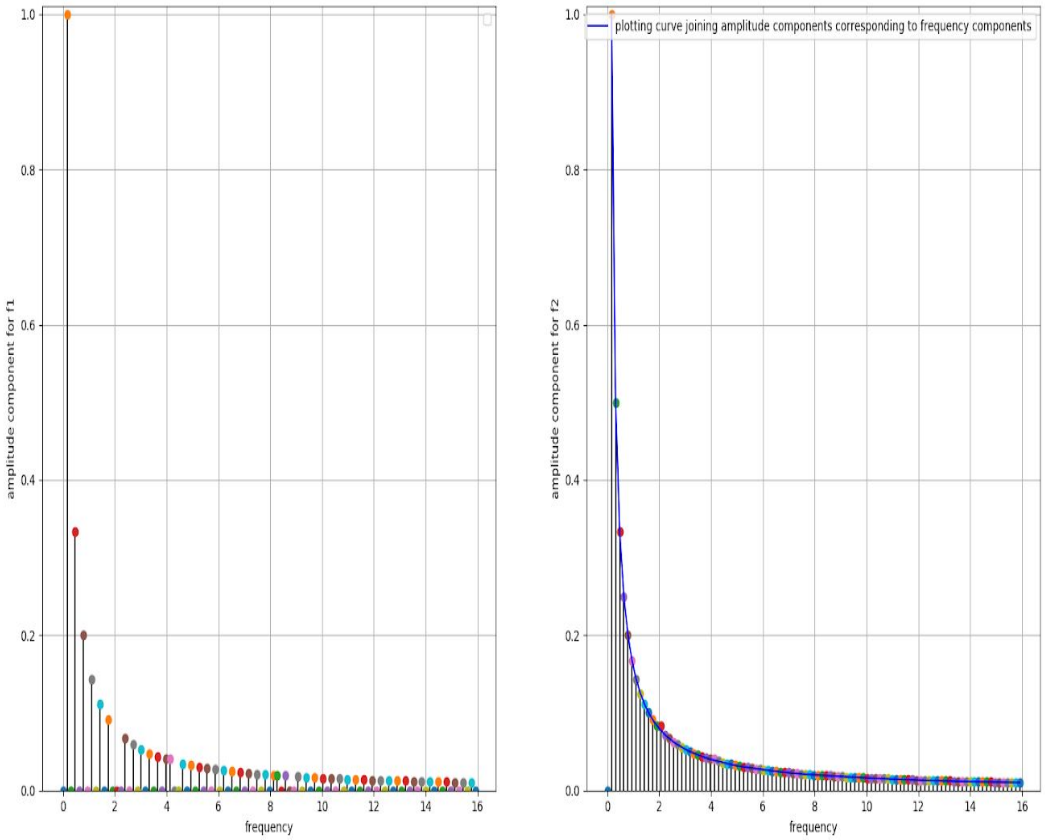


Q.1_(c).py (Part (c) in Question.1) :

```
import numpy as np
import matplotlib.pyplot as plt
import cmath
from pylab import *
##FOR f1:
ak = [0]*101
ak_new = [0]*51
for k in range(0,101):
        if(k%2 == 0):
                ak[k] = 0
        else:
                ak[k] = 1/k
#Above represents the amplitude components of f1.
frequency = [0]*101
for i in range(0,101):
        frequency[i] = i/(2*np.pi)  #Represents the frequency harmonics of f1.
subplot(1,2,1)
for xc in frequency:
        j = np.int(xc*2*pi)
        plt.plot(xc,ak[j],'o')
        plt.vlines(xc, ymin=0, ymax=ak[j])#plotting the amplitude components v/s
frequency harmonics.

plt.ylim(0,1.01)
plt.xlabel('frequency')
plt.ylabel('amplitude component for f1')
plt.legend()
plt.grid()
##############################################################################
###################

##FOR f2:

ak1 = [0]*101
for k1 in range(1,101):
        ak1[k1] = 1/k1
#Above represents the amplitude components of f2.
```

```
frequency1 = [0]*101
for i1 in range(0,101):
	frequency1[i1] = i1/(2*np.pi)  #Represents the frequency harmonics of f2.
subplot(1,2,2)
for xc1 in frequency1:
	j1 = np.int(xc1*2*pi)
	plt.plot(xc1,ak1[j1],'o')
	plt.vlines(xc1, ymin=0, ymax=ak1[j1])#plotting the amplitude components v/s
frequency harmonics.
frequency1_new = [0]*100
ak1_new = [0]*100
for i1 in range(0,100):
	frequency1_new[i1] = (i1+1)/(2*np.pi)
	ak1_new[i1] = 1/(i1+1)
plt.plot(frequency1_new,ak1_new,'b',label='plotting curve joining amplitude components
corresponding to frequency components')#plotting line curve joining amplitude
components corresponding to all frequency components.
plt.ylim(0,1.01)
plt.xlabel('frequency')
plt.ylabel('amplitude component for f2')
plt.legend()
plt.grid()
plt.show()
```

Figure Obtained after running the code:



Q.1_(d).py (Part (d) in Question.1) :

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import cmath
t = np.linspace(-50,50,1000)

def Fourier_cal(N):
	y = 0
	z = complex(0,1)
	for n in range(1,N,2):
		y = y + (1/(n**2))*np.cos(n*t)
	return y
```

```
y1 = np.cos(t)
y2 = Fourier_cal(5)
y3 = Fourier_cal(20)

subplot(3,1,1)
plt.plot(t,y1)
plt.xlabel('t')
plt.ylabel('sin(t)')
plt.grid()

subplot(3,1,2)
plt.plot(t,y2)
plt.xlabel('t')
plt.ylabel('Fourier approximation when N = 5')
plt.grid()

subplot(3,1,3)
plt.plot(t,y3)
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier approximation when N = 20')
plt.show()
##here we can modify the coefficients of cosine function  (( for n in range(1,N,2):  y = y
+ (1/(n**2))*np.cos(n*t) return y))   in the summation as 1/(n^2),then only we get the
symmetric triangular function.
```

Q.1 (d) $\Rightarrow$

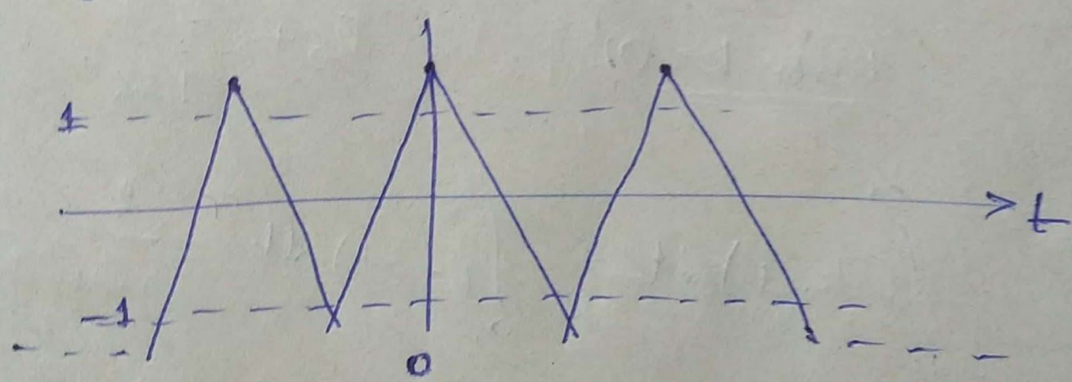
If we want to get the symmetric function about y-axis (even function) for e.g Triangle function, then

$$\boxed{f(t) = \sum_{n=1,3,5}^{N} \frac{1}{n^2}\cos(nt)}$$

i.e $a_n \propto \frac{1}{n^2}$

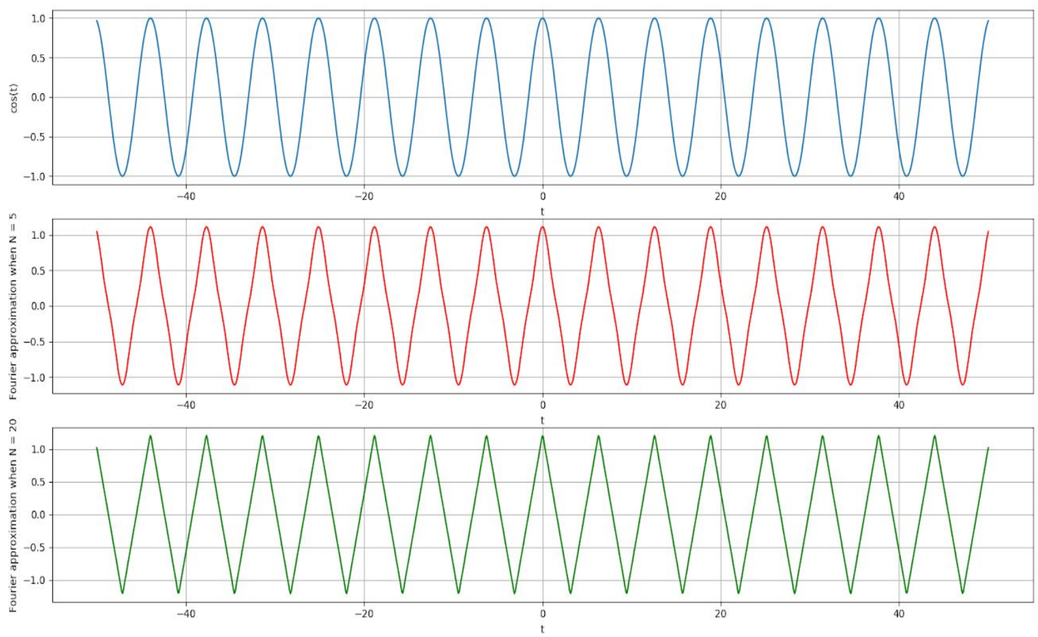
(for Triangular waveform)

After writing program about this function, f(t), we get a symmetric function about y-axis (even function)



(This is obtained after selecting sufficiently high value of N.)

Figure Obtained after running the code:



Q.1 (e) ⇒

I have ploted the graphs if we require the desired value of A and D.

Basically, Duty cycle $= \dfrac{\text{Pulse width}}{T \text{(Time period)}}$

Below are the two programs which will basically shows how we can get desired D and A values by just changing the fourier Series Coefficients.

For $f_1(t) \Rightarrow$

$$f_1(t) = \sum_{\substack{\cancel{n=1,2,3\ldots} \\ n=-N}}^{N} \left(\frac{A}{j\pi n}\left(1 - e^{-j2\pi n D}\right)\right) e^{jnt} \quad \rightarrow \underline{n \neq 0}$$

and for $f_2(t) \Rightarrow$

$$f_2(t) = \sum_{n=-N}^{N} \frac{-A}{4\pi^2}\left(\frac{4\pi j}{n} + \left(\frac{2D-1}{D(1-D)n^2}\right)\left(1 - e^{-j2\pi n D}\right)\right) e^{jnt}$$

$$n \neq 0$$

Q.1_(e)_changing_DutyCycle_and_Amplitude_of_f1_code.py

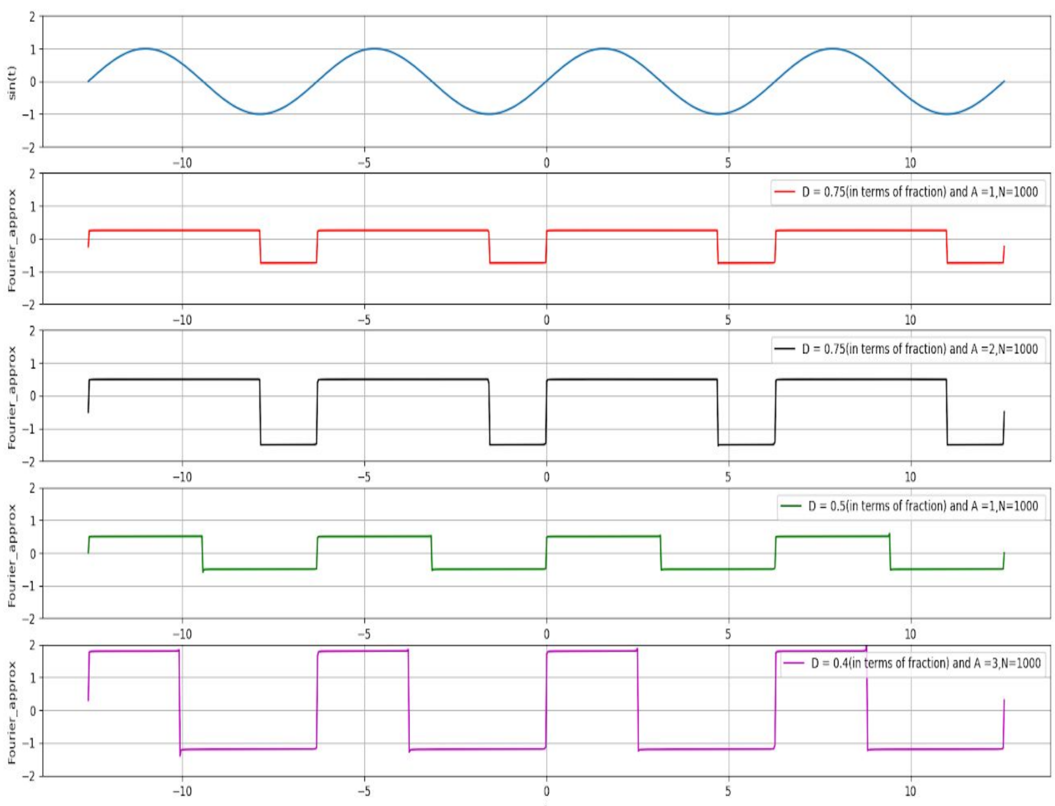
```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import cmath
t = np.linspace(-4*np.pi,4*np.pi,1000)

def Fourier_cal(N,A,D):
        y = 0
        z = complex(0,1)
        for n in range(-N,N,1):
                if(n!=0):
                        y = y + ((A/(np.pi*z*n))*(1-np.exp(-z*n*(2*np.pi*D))))*np.exp(z*n*t)
#By changing the coefficient of the fourier series,we can easily modify the waveform
according to Duty cycle and Amplitude.i.e Cn =
((A/(np.pi*z*n))*(1-np.exp(-z*n*(2*np.pi*D))))
        return y
y1 = np.sin(t)
```

```
y2 = Fourier_cal(1000,1,3/4)# A =1 , D = 0.75(in terms of fraction)
y3 = Fourier_cal(1000,2,3/4)# A =2 , D = 0.75(in terms of fraction)
y4 = Fourier_cal(1000,1,1/2)# A =1 ,D = 0.5(in terms of fraction)
y5 = Fourier_cal(1000,3,0.4)# A =3 ,D = 0.4(in terms of fraction)

subplot(5,1,1)
plt.plot(t,y1)
plt.xlabel('t')
plt.ylabel('sin(t)')
plt.grid()



subplot(5,1,2)
plt.plot(t,y2,'r',label='D = 0.75(in terms of fraction) and A =1,N=1000 ')
plt.xlabel('t')
plt.ylabel('Fourier_approx')
plt.legend()
plt.grid()



subplot(5,1,3)
plt.plot(t,y3,'k',label='D = 0.75(in terms of fraction) and A =2,N=1000 ')
plt.xlabel('t')
plt.ylabel('Fourier_approx')
plt.legend()
plt.grid()


subplot(5,1,4)
plt.plot(t,y4,'g',label='D = 0.5(in terms of fraction) and A =1,N=1000 ')
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier_approx')
plt.legend()
```

```
subplot(5,1,5)
plt.plot(t,y5,'m',label='D = 0.4(in terms of fraction) and A =3,N=1000')
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier_approx')
plt.legend()


plt.show()
```

####Similarly by changing the fourier coefficients for f2(t) as mentioned above , we can get the waveform according to the required need of Duty cycle and amplitude.

Figure Obtained after running the code:

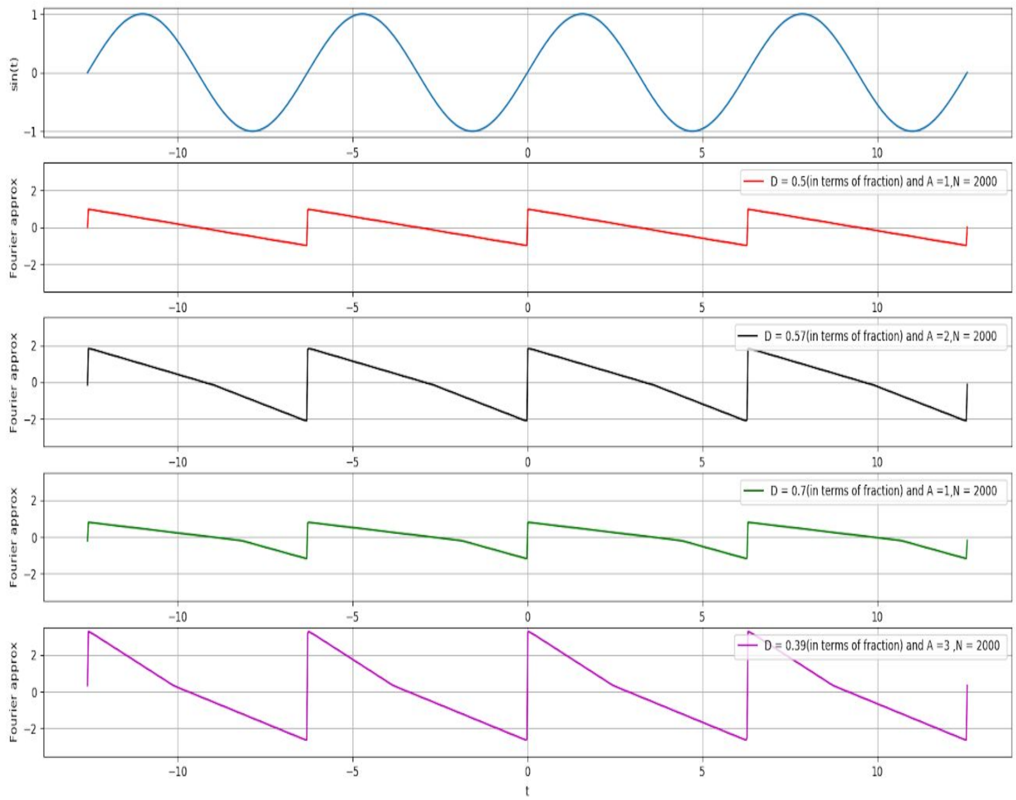


Q.1_(e)_changing_DutyCycle_and_Amplitude_of_f2_code.py

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import cmath
t = np.linspace(-4*np.pi,4*np.pi,1000)

def Fourier_cal(N,A,D):
	y = 0
	z = complex(0,1)
	for n in range(-N,N,1):
		if(n!=0):
			y = y +
((-A/((2*np.pi)**2))*(((4*np.pi*z)/n)+((((2*D)-1)/(D*(1-D)*(n**2)))*(1-np.exp(-z*2*np.pi*n*
D)))))*np.exp(z*n*t) #By changing the coefficient of the fourier series,we can easily
modify the waveform according to Duty cycle and Amplitude.
	return y
y1 = np.sin(t)
y2 = Fourier_cal(2000,1,0.5)# A =1 , D = 0.5(in terms of fraction)
y3 = Fourier_cal(2000,2,0.57)# A =2 , D = 0.57(in terms of fraction)
y4 = Fourier_cal(2000,1,0.7)# A =1 ,D = 0.7(in terms of fraction)
y5 = Fourier_cal(2000,3,0.39)# A =3 ,D = 0.39(in terms of fraction)
subplot(5,1,1)
plt.plot(t,y1)
plt.xlabel('t')
plt.ylabel('sin(t)')
plt.grid()

subplot(5,1,2)
plt.plot(t,y2,'r',label='D = 0.5(in terms of fraction) and A =1,N = 2000  ')
plt.xlabel('t')
plt.ylabel('Fourier approx')
plt.legend()
plt.grid()
plt.ylim(-3.5,3.5)

subplot(5,1,3)
plt.plot(t,y3,'k',label='D = 0.57(in terms of fraction) and A =2,N = 2000  ')
```

```
plt.xlabel('t')
plt.ylabel('Fourier approx')
plt.legend()
plt.grid()
plt.ylim(-3.5,3.5)

subplot(5,1,4)
plt.plot(t,y4,'g',label='D = 0.7(in terms of fraction) and A =1,N = 2000  ')
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier approx')
plt.legend()
plt.ylim(-3.5,3.5)

subplot(5,1,5)
plt.plot(t,y5,'m',label='D = 0.39(in terms of fraction) and A =3 ,N = 2000 ')
plt.grid()
plt.xlabel('t')
plt.ylabel('Fourier approx')
plt.legend()
plt.ylim(-3.5,3.5)
plt.show()
```

Figure Obtained after running the code:



In this way,we can modify the summation such that duty cycle can adjusted to a desired duty cycle D, and amplitude A for both f1 and f2 functions.