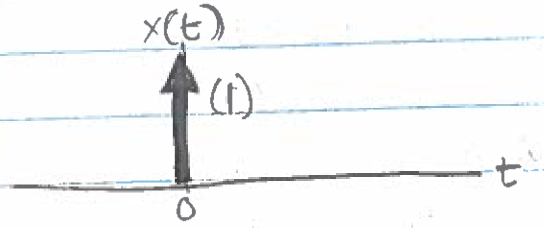
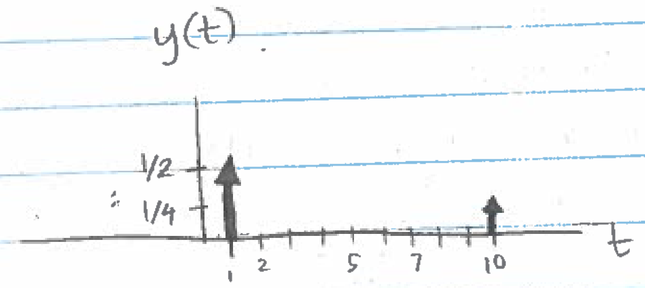
ALL code probs are at the end
(3b and 4b)

1. An impulse contains all frequencies, because being such a quick signal, with a very quick change over a very small time, it has many high frequency components. Since it has so many frequencies, we are able to characterize a system using just an impulse and looking at the impulse response, which is how a system responds to an impulse. The impulse response is in the time domain so if you take the Fourier transform you get the transfer function and now you know what the system does to any frequency. So you can put a violin recording with certain frequencies and see what the system will do to it using the transfer function.

2. $y(t) = \frac{1}{2}x(t-1) + \frac{1}{4}x(t-10)$




~~If I~~ put an impulse of 7 into the system, it will output an impulse half the amplitude 1 second later, and another impulse 1/4 of the amplitude 10 seconds later.

So, it seems reasonable to describe this as an echo channel.

if $x(t)$ is an impulse $\delta(t)$, the impulse response $h(t)$ is:

$$h(t) = \frac{1}{2}\delta(t-1) + \frac{1}{4}\delta(t-10)$$

3a) First, find $C_k$

$$C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\, e^{-j\frac{2\pi}{T}kt}\, dt$$

$x(t)=1$ for $t = -T/4$ to $T/4$

$x(t)=0$ for $t = -T/2$ to $-T/4$

$x($

$$C_k = 0 \quad \begin{cases} t = -T/2 \text{ to } -T/4 \\ t = -T/4 \text{ to } T/2 \end{cases}$$

amplitudes

$$C_k = \frac{1}{T}\int_{-T/4}^{T/4} x(t)\, e^{-j\frac{2\pi}{T}kt}\, dt$$

~~Fourier Series??~~

$$C_k = \frac{1}{T}\int_{-T/4}^{T/4} e^{-j\frac{2\pi}{T}kt}\, dt$$

plugging in $x(t)=1$ for $-T/4$ to $T/4$

$$= \frac{1}{T}\left[\frac{1}{-j\frac{2\pi}{T}k} e^{-j\frac{2\pi}{T}kt}\right]_{-T/4}^{T/4}$$

$$= \frac{1}{T}\left(\frac{1}{-j\frac{2\pi}{T}k} e^{-j\frac{2\pi}{T}k\left(\frac{T}{4}\right)} - \frac{1}{-j\frac{2\pi}{T}k} e^{-j\frac{2\pi}{T}k\left(-\frac{T}{4}\right)}\right)$$

$$= \frac{1}{T}\left(\frac{1}{-j\frac{2\pi}{T}k} e^{-j\pi\frac{k}{2}} - \frac{1}{-j\frac{2\pi}{T}k} e^{j\pi\frac{k}{2}}\right)$$

$$= -\frac{1}{j2\pi k} e^{-j\pi\frac{k}{2}} + \frac{1}{j2\pi k} e^{j\pi\frac{k}{2}})$$

$$\frac{1}{\pi k}\left(\frac{1}{2j} e^{j\pi\frac{k}{2}} - \frac{1}{2j} e^{-j\pi\frac{k}{2}}\right)$$

$$\frac{1}{1/2} \quad \frac{1}{\pi k}\left[\sin\left(\pi\frac{k}{2}\right)\right]$$

$$\frac{\sin\left(\pi\frac{k}{2}\right)}{\pi\, k/2} = \frac{1}{2}\operatorname{sinc}\left(\frac{k}{2}\right)$$

this is $C_k$

Now need to write Fourier Series using $C_k$

3a cont

$C_k = \text{sinc}(\frac{k}{2}) \quad \therefore$

$$\tilde{x}_K(t) = \sum_{k=-K}^{K} C_k e^{j\frac{2\pi}{T}kt}$$

The 1/2 comes from being off by a factor of 2 when trying to make $C_k$ look like sinc.

$$\boxed{\tilde{x}_K(t) = \sum_{k=-K}^{K} \frac{1}{2}\text{sinc}(\tfrac{k}{2}) e^{j\frac{2\pi}{T}kt}}$$

~~4a~~ $\cancel{y(t) = x(t - T_1)}$

~~Let~~ $\cancel{w = t - T_1}$ $\quad \cancel{t = w + T_1}$

$\cancel{y(t) = x(w)}$

$$\cancel{C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t) e^{-j\frac{2\pi}{T}kt}\, dt}$$

~~Let~~ $\cancel{w = t - T_1}$

$$\cancel{C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(w+T_1) e^{-j\frac{2\pi}{T}k(w+T_1)}\, dw}$$

3c. The discontinuities from 0 to 1 and 1 to 0 look a little weird especially with just a few terms of approximation. They have bumps near the area of the discontinuity. This is because a true square wave goes from 0 to 1 and 1 to 0 in an infinitesimally small amount of time, which corresponds to high frequencies. So if you are approximating it with only a few terms (ie low frequencies), it's going to look weird. But as K, the number of terms goes to infinity, the square wave approximation looks better and better because we are including these high frequency components.

4a. $\underbrace{C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)e^{-j\frac{2\pi}{T}kt}\,dt}$

since $y(t) = x(t-T_1)$,

let $w = t - T_1$, so $t = w + T_1$

$\frac{1}{T}\int_{-T/2}^{T/2} x(w+T_1)e^{-j\frac{2\pi}{T}k(w+T_1)}\,dw$

$\frac{1}{T}\int_{-T/2}^{T/2} x(w+T_1)\,e^{-j\frac{2\pi}{T}kw}\,e^{-j\frac{2\pi}{T}kT_1}\,dw$

$\boxed{C_k \text{ for } y(t) = C_k e^{-j\frac{2\pi}{T}kT_1}}$

multiply by a factor of $e^{-j\frac{2\pi}{T}kT_1}$ to offset by $T_1$.

In [12]:
```
import numpy as np
import matplotlib.pyplot as mplib
% matplotlib inline
```

## Fourier Series Function

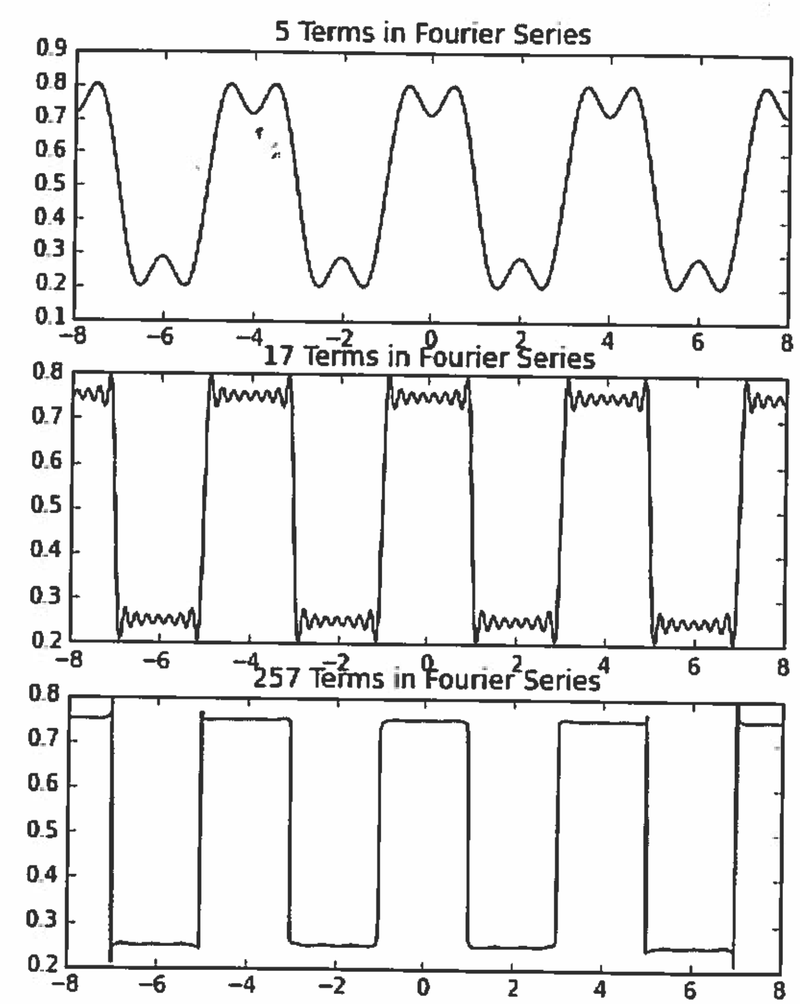
In [21]:
```
def fourier_series(t,K):
    T=4
    x=0
    for k in range(len(K)):
            ##########################################################################################################
            ##  change the following line to provide the Fourier series coefficients for the square wave
            Coeff = .5*np.sinc(k/2.0)

            x = x + Coeff*np.exp(1j*2.0*np.pi/T*k*t)
    return x
```

In [26]:
```
K_5 = np.arange(-2,2,1)
t = np.linspace(-8,8,1000)

K_17 = np.arange(-8,8,1)
t = np.linspace(-8,8,1000)

K_257 = np.arange(-128,128,1)
t = np.linspace(-8,8,1000)
```

## 3b. Fourier series representation of a square wave with 5 terms, 17 terms and 257 terms.

In [38]:
```
mplib.subplot(3,1,1)
mplib.subplots_adjust(left=None, bottom=2, right=None, top=3.5, wspace
=None, hspace=None)
mplib.plot(t, fourier_series(t,K_5))
mplib.title('5 Terms in Fourier Series')


mplib.subplot(3,1,2)
mplib.plot(t, fourier_series(t,K_17))
mplib.title('17 Terms in Fourier Series')


mplib.subplot(3,1,3)
mplib.plot(t, fourier_series(t,K_257))
mplib.title('257 Terms in Fourier Series')
```

Out[38]: <matplotlib.text.Text at 0x7f4be540ee90>



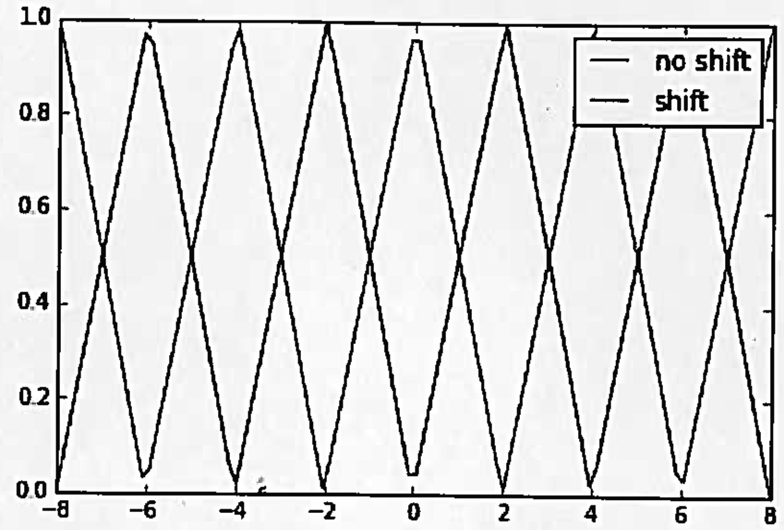

In [0]:

**4b. Triangle waves with offsets. Multiplying by the coefficient found in part a.**

In [46]:
```
def fs_triangle(ts, T1, M=100, T=4):
    # computes a fourier series representation of a triangle wave
    # with M terms in the Fourier series approximation
    # if M is odd, terms -(M-1)/2 -> (M-1)/2 are used
    # if M is even terms -M/2 -> M/2-1 are used

    # create an array to store the signal
    x = np.zeros(len(ts))

    # if M is even
    if np.mod(M,2) ==0:
        for k in range(-int(M/2), int(M/2)):
            # if n is odd compute the coefficients
            if np.mod(k, 2)==1:
                Coeff = -2/((np.pi)**2*(k**2))
            if np.mod(k,2)==0:
                Coeff = 0
            if k == 0:
                Coeff = 0.5
            x = x + (Coeff*np.exp(1j*2*np.pi/T*k*T1))*np.exp(1j*2*np.p
i/T*k*ts)

    # if M is odd
    if np.mod(M,2) == 1:
        for k in range(-int((M-1)/2), int((M-1)/2)+1):
         # if n is odd compute the coefficients
            if np.mod(k, 2)==1:
                Coeff = -2/((np.pi)**2*(k**2))
            if np.mod(k,2)==0:
                Coeff = 0
            if k == 0:
                Coeff = 0.5
            x = x + (Coeff*np.exp(1j*2*np.pi/T*k*T1))*np.exp(1j*2*np.p
i/T*k*ts)

    return x
```

In [50]:
```
ts = np.linspace(-8,8,100)
triangle_no_shift, = mplib.plot(ts, fs_triangle(ts,0))
triangle_shift, = mplib.plot(ts, fs_triangle(ts,2))
mplib.legend([triangle_no_shift, triangle_shift], ['no shift', 'shift'
])
```

Out[50]: <matplotlib.legend.Legend at 0x7f4be4e64650>




The green wave was made to match figure 2 from the problem set. I had to modify the coefficient from 4a so I could shift it and center the peak at 0.

In [ ]: