```python
# Author: Ramesh Pai
# Affiliation: 201104047, TE-E&TC Engg, Sem.5, 2021-22, GCE.

# importing python modules
import matplotlib.pyplot as plt
import numpy as np
from scipy.fft import fft, fftshift, ifft, ifftshift

# vm1: Amplitude of Message Signal 1
# vm2: Amplitude of Message Signal 2
# vc: Amplitude of Carrier Signal
# fm1: fequency(Hz) of Message Signal 1
# fm2: fequency(Hz) of Message Signal 2
# fc: frequency(Hz) of Carrier Signal

def values(vm1 , vm2, vc, fm1, fm2, fc):

    fs = 60000 #sampling frequency
    dt = 1/fs #sample time interval or time-steps for time-domain signal
    t = np.arange(0, 0.2, dt) #time indices for time-domain signal
    n = np.size(t) #number of samples
    df = fs/n #frquency interval or frequency-steps for frequency-spectrum
    f = np.arange(-fs/2, fs/2, df) #frequency indices for frquency-spectrum

    # plot1: Two-tone Modulating Signal v/s Time
    vtm = vm1*np.sin(2*np.pi*fm1*t) + vm2*np.sin(2*np.pi*fm2*t) #Two-tone Message Signal
    plt.subplot(3, 3, 1)
    plt.plot(t, vtm)
    plt.title("Message Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("vtm(Volts)")

    # plot2: Carrier Signal v/s Time
    vtc = vc*np.sin(2*np.pi*fc*t) #Carrier Signal
    plt.subplot(3, 3 ,2)
    plt.plot(t, vtc)
    plt.title("Carrier Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("vtc(Volts)")

    # plot3: AM Signal v/s Time
    v_am = np.sin(2*np.pi*fc*t)*(vc + (vm1*np.sin(2*np.pi*fm1*t) + vm2*np.sin(2*np.pi*fm2*t))) #AM Modulated Signal
    plt.subplot(3, 3, 3)
    plt.plot(t, v_am)
    plt.title("Modulated Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v_am(Volts)")

    #plot4: DFT(magnitude) of AM Signal v/s Frequency
    xf1 = fftshift(fft(v_am)) #FFT of Modulated Signal(Complex in nature).
    plt.subplot(3, 3, 4)
    plt.plot(f, abs(xf1)/n) #PLotting frequency indices v/s Normalised magnitude of FFT Modulated signal
    plt.xlim(400, -400)
    plt.title("AM modulated frequency Spectrum", loc='left')
    plt.xlabel("frequency(Hz)", loc='right')
    plt.ylabel("Magnitude")
```

```python

    #  plot5: Demodulated Signal v/s Time
    v_am2 = v_am*np.sin(2*np.pi*fc*t) #Multiplying Modulated signal with Carrier Signal to get Demodulated Signal
    plt.subplot(3, 3, 5)
    plt.plot(t, v_am2)
    plt.title("Demodulated AM Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("v_am2(Volts)")

    xf2 = fftshift(fft(v_am2)) #FFT of Demodulated Signal(Complex in nature)
    plt.subplot(3, 3, 6)
    plt.plot(f, abs(xf2)/n) #PLotting frequency indices v/s Normalised magnitude of FFT Demodulated signal
    plt.xlim(400, -400)
    plt.title("Demodulated frequency Spectrum", loc='left')
    plt.xlabel("frequency(Hz)", loc='right')
    plt.ylabel("Magnitude")

    # Designing Filter for aquiring Original Two-tone Signal.
    l1 = [] #List having array of 0's and 1's

    for x in f:
        if x < max(fm1, fm2)+20 and x > min(-fm1, -fm2)-20:
            x = 1
            l1.append(x) #Assigning 1 to frequencies below Cutoff
        else:
            x = 0
            l1.append(x) #Assigning 0 to frequencies above Cutoff

    filteredSpectrum = xf2*l1 #Multiplying Filter with FFT demodulated Signal to Aquire Original Two-tone signal

    vr = ifft(ifftshift(filteredSpectrum)) #Inverse FFT to get Original Two-tone signal(time-domain)
    plt.subplot(3, 3, 7)
    plt.plot(t, abs(vr)) #Removing Complex values
    plt.title("Demodulated Two-tone Signal", loc='left')
    plt.xlabel("t(sec)", loc='right')
    plt.ylabel("vtm(Volts)")

    plt.show()

values(1, 1, 2, 20, 50, 150) #Assigning values to defined parameters


```