Discrete Time Fourier Transform (DTFT)

Q1. Compute the DTFT (magnitude and phase) of the following (use *scipy.signal.freqz*). Plot from $\omega = -2\pi$ to 2π. Observe the symmetries and relations between the spectra.

https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.signal.freqz.html

1. $r[n] = u[n] - u[n-5]$
2. $r[n-7]$
3. $r[n+4]$
4. $r[-n]$
5. $(-1)^n r[n]$

Inference

What is the period of the DTFT ?

There are two different discontinuities in the phase spectrum. Identify and explain why it is happeneing?

CODE

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
```

In [2]:
```
N = 4096
omega = 2 * np.pi
n = 15
x = np.arange(-n, n + 1)
x
```

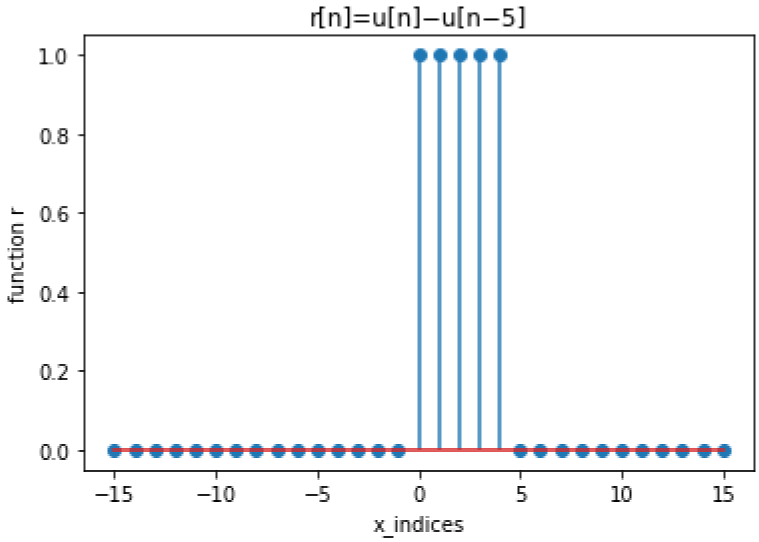
Out[2]:
```
array([-15, -14, -13, -12, -11, -10,  -9,  -8,  -7,  -6,  -5,  -4,  -3,
        -2,  -1,   0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
        11,  12,  13,  14,  15])
```

1) $r[n] = u[n] - u[n-5]$

In [3]:
```
n0 = 0
```

In [4]:
```
r = np.zeros(x.size)
r[n + n0:n + n0 + 5] = 1
plt.stem(x, r)
plt.xlabel('x_indices')
plt.ylabel('function r')
plt.title('r[n]=u[n]-u[n-5]')
```

Out[4]:
```
Text(0.5, 1.0, 'r[n]=u[n]-u[n-5]')
```



In [5]:

```
a = np.ones(1)
b = r[n:]
a, b
```

Out[5]:

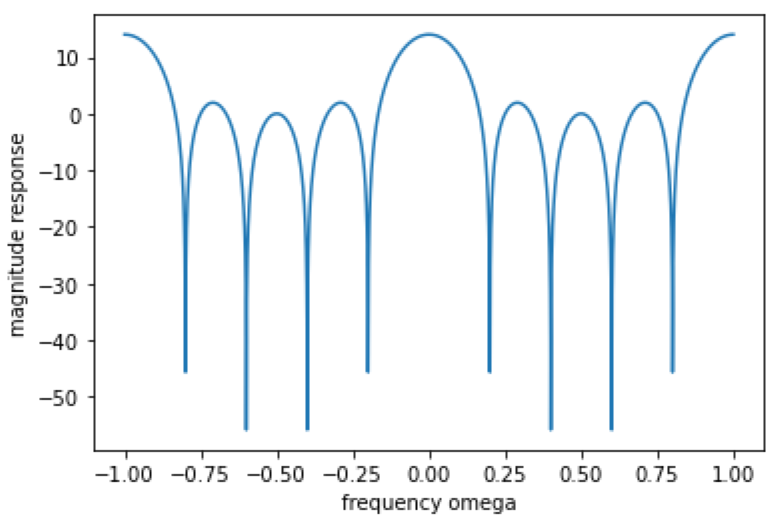
```
(array([1.]),
 array([1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]))
```

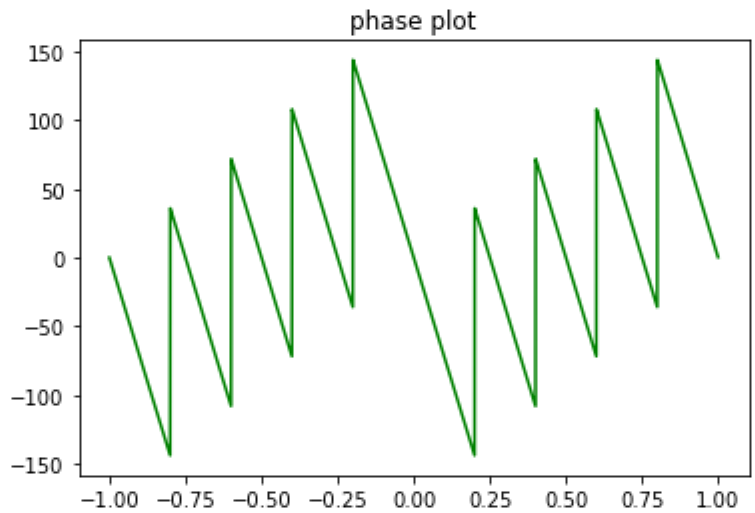
In [6]:

```
def plot(r, a, b):
    c = np.arange(-omega, omega, 2 * omega / N)
    w1, h1 = sig.freqz(b, a, c)
    #     magnitude
    h1db = 20 * np.log10(abs(h1))
    plt.plot(w1 / (2 * np.pi), h1db)
    plt.xlabel('frequency omega')
    plt.ylabel('magnitude response')
    plt.show()
    #     phase
    angles = np.angle(h1, deg=True)
    plt.plot(w1 / (2 * np.pi), angles, 'g')
    plt.title('phase plot')
    plt.show()
```

In [7]:

```
plot(r, a, b)
```





INFERENCE

The period of DTFT is 2pi

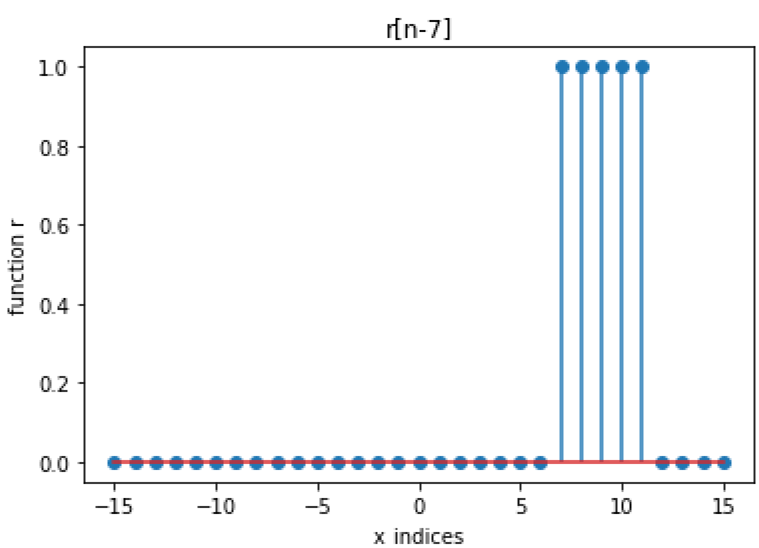
As we know that DTFT is 2pi periodic (-pi to pi for instance as example), the phase plot would be continous at that interval.But as soon as the DTFT moves outside the -pi to +pi range and when it starts the next cycle , the phase plot would experience a discontionuity at that point eventhough mathematically if it had a continous function of omega(phase is computed modulo 2pi)...it breaks and starts a phase plot similar to one before when DTFT was in 1st cycle.

2) $r[n-7]$

In [8]:
```
n0 = 7
```

In [9]:
```
r = np.zeros(x.size)
r[n + n0:n + n0 + 5] = 1
plt.stem(x, r)
plt.xlabel('x_indices')
plt.ylabel('function r')
plt.title('r[n-7]')
```

Out[9]:
```
Text(0.5, 1.0, 'r[n-7]')
```

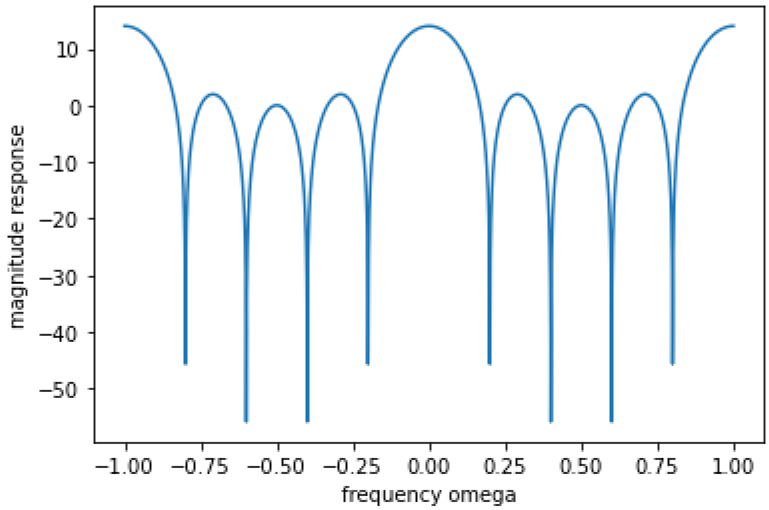


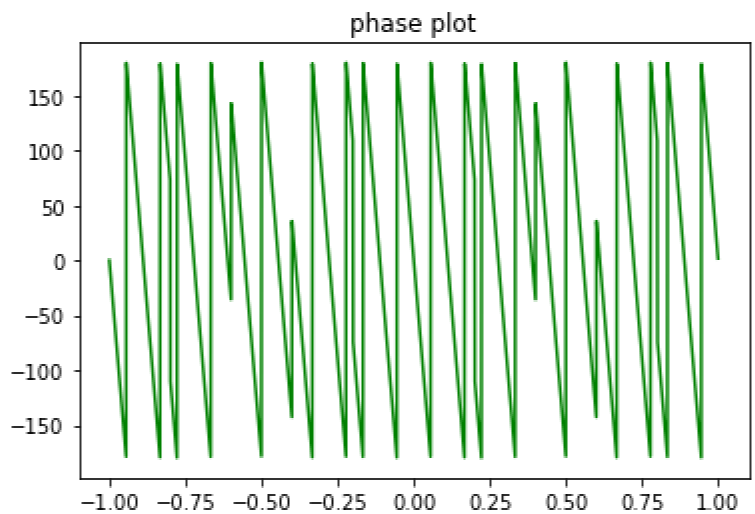
In [10]:
```
a = np.ones(1)
b = r[n:]
```

```
a, b
```

Out[10]:
```
(array([1.]),
 array([0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 0.]))
```

In [11]:
```
plot(r, a, b)
```



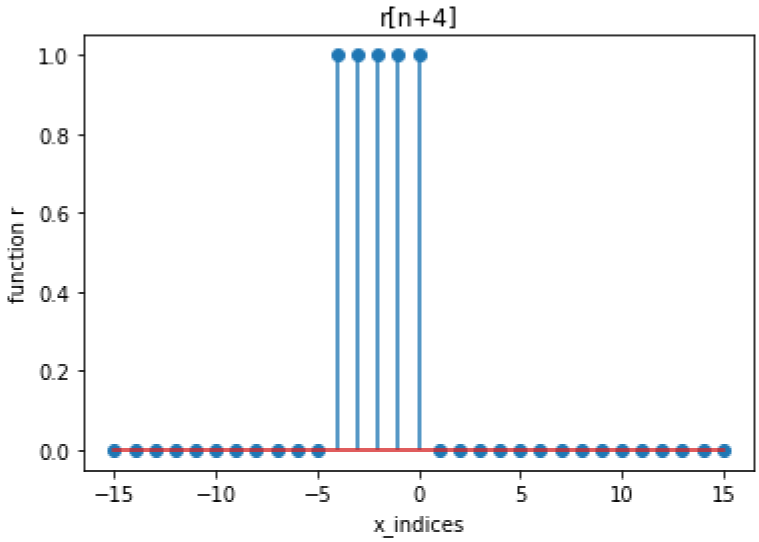


3) $r[n+4]$

In [12]:
```
n0 = -4
```

In [13]:
```
r = np.zeros(x.size)
r[n + n0:n + n0 + 5] = 1
plt.stem(x, r)
plt.xlabel('x_indices')
plt.ylabel('function r')
plt.title('r[n+4]')
```

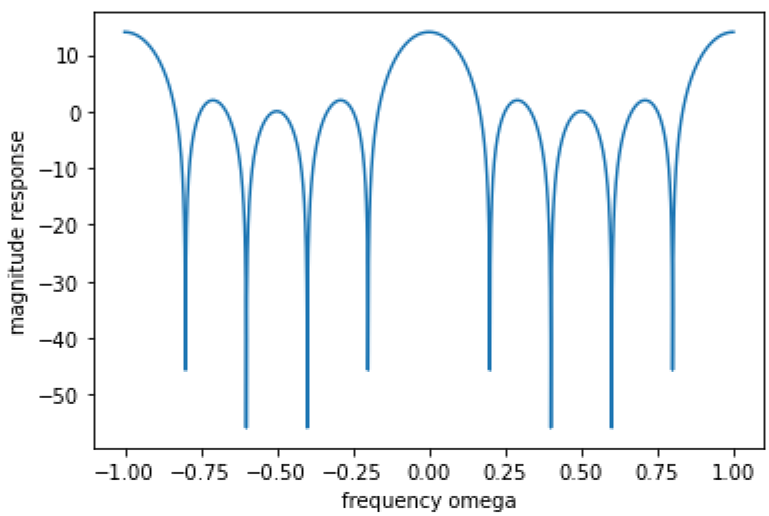
Out[13]:
```
Text(0.5, 1.0, 'r[n+4]')
```

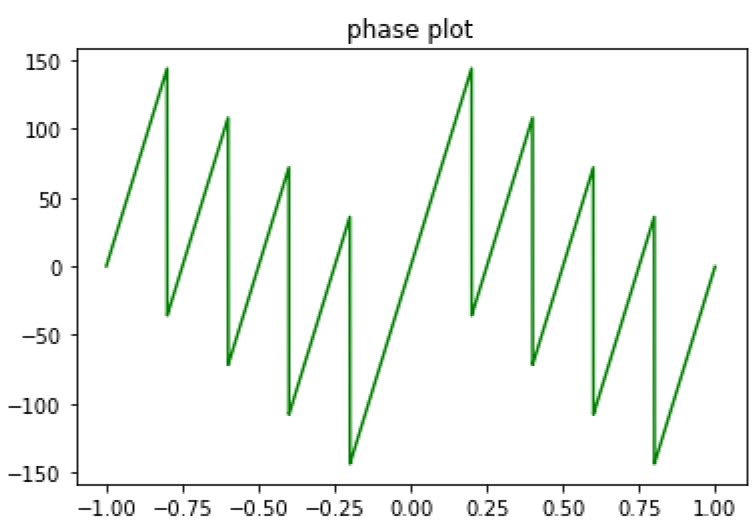


In [14]:
```
a = np.zeros(5)
a[4] = 1
b = np.ones(5)
a, b
```

Out[14]:
```
(array([0., 0., 0., 0., 1.]), array([1., 1., 1., 1., 1.]))
```

In [15]:
```
plot(r, a, b)
```



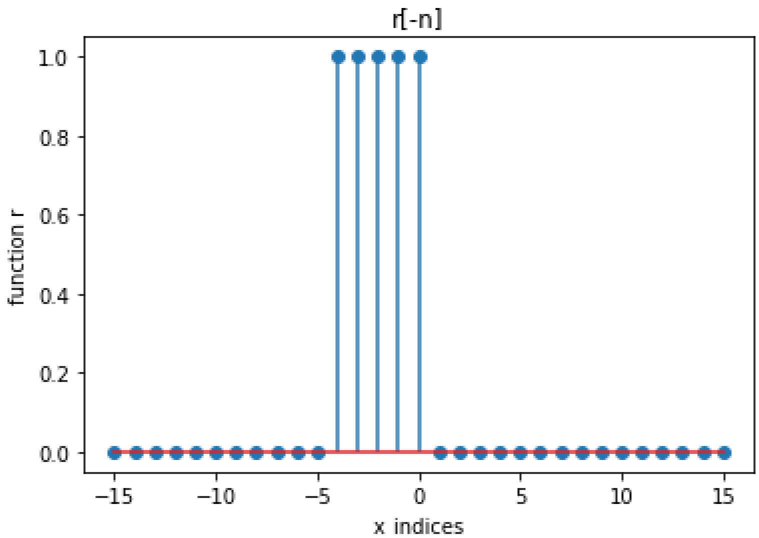


4) $r[-n]$

In [16]:
```
n0 = 0
```

In [17]:
```
r = np.zeros(x.size)
r[n + n0:n + n0 + 5] = 1
r = r[::-1]
plt.stem(x, r)
plt.xlabel('x_indices')
plt.ylabel('function r')
plt.title('r[-n]')
```

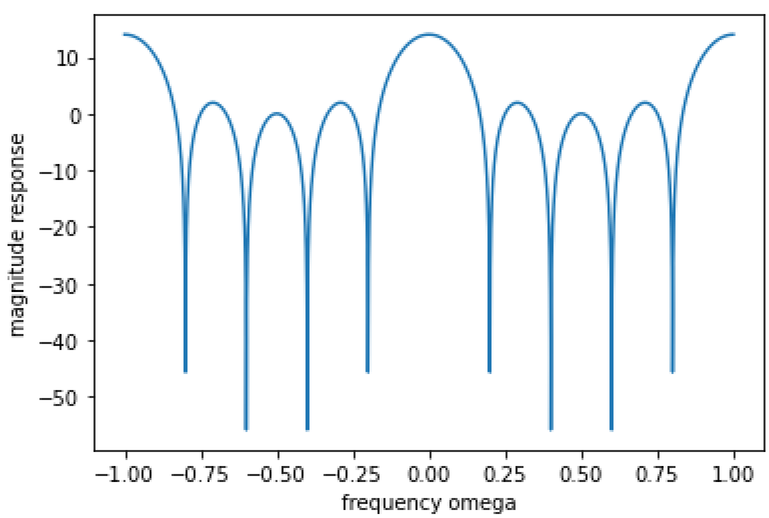
Out[17]:
```
Text(0.5, 1.0, 'r[-n]')
```

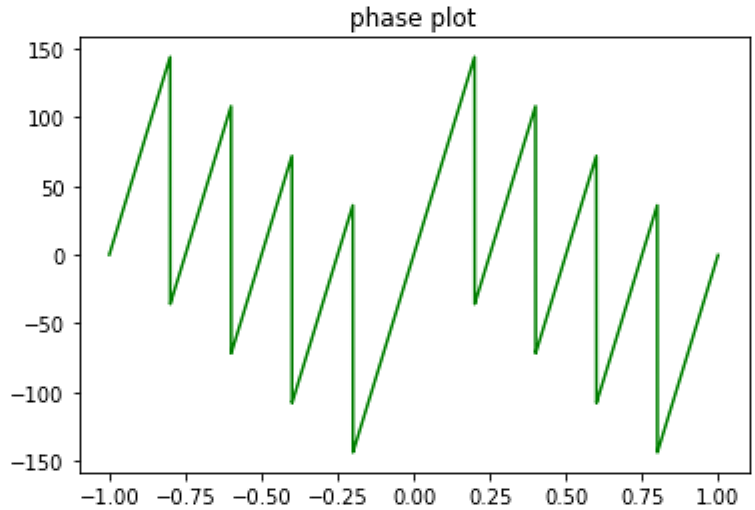


In [18]:
```
a = np.zeros(5)
a[4] = 1
b = np.ones(5)
a, b
```

Out[18]:
```
(array([0., 0., 0., 0., 1.]), array([1., 1., 1., 1., 1.]))
```

In [19]:
```
plot(r, a, b)
```



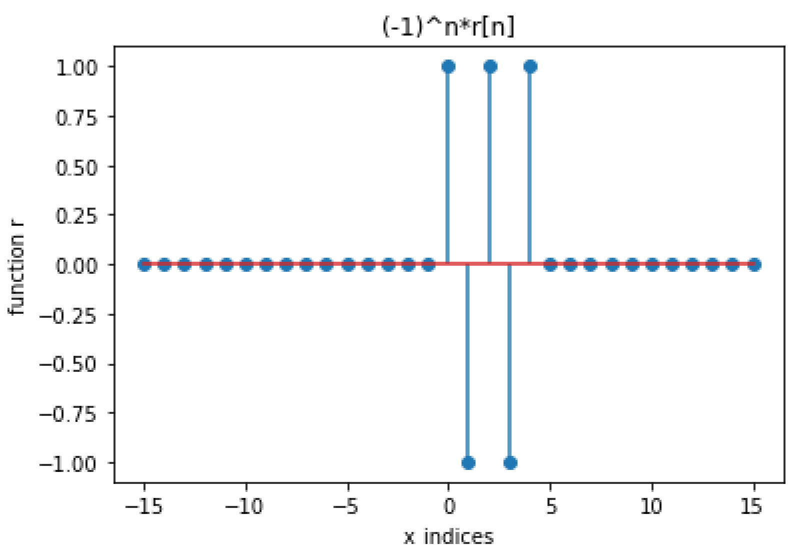


5) $(-1)^n r[n]$

In [20]:
```
n0 = 0
```

In [21]:
```
r = np.zeros(x.size)
r[n + n0:n + n0 + 5] = 1
r[x % 2 == 1] *= -1
plt.stem(x, r)
plt.xlabel('x_indices')
plt.ylabel('function r')
plt.title('(-1)^n*r[n]')
```

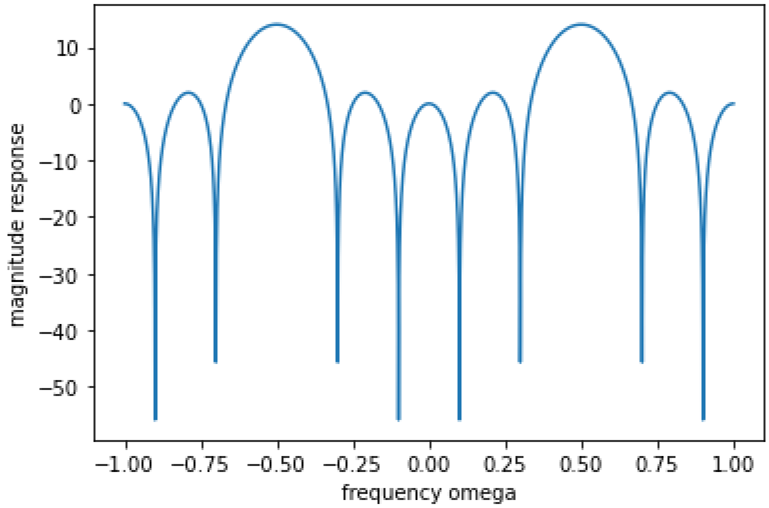
Out[21]:
```
Text(0.5, 1.0, '(-1)^n*r[n]')
```

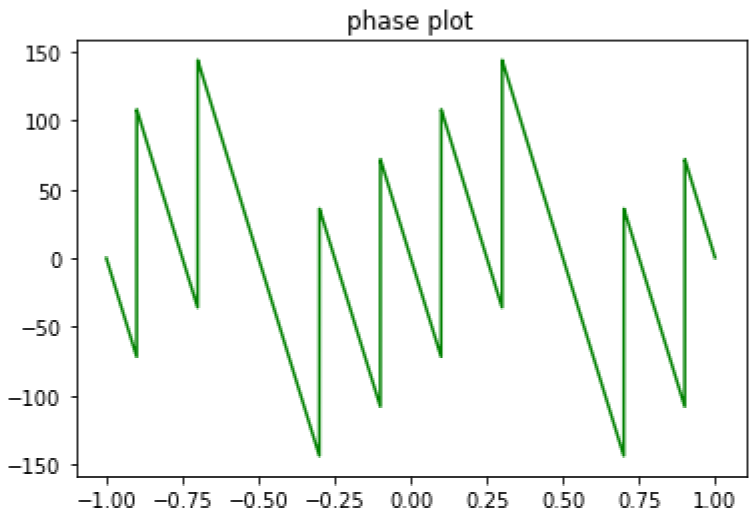


In [22]:
```
a = np.ones(1)
b = r[n:]
a, b
```

Out[22]:
```
(array([1.]),
 array([ 1., -1.,  1., -1.,  1., -0.,  0., -0.,  0., -0.,  0., -0.,  0.,
        -0.,  0., -0.]))
```

In [23]:
```
plot(r, a, b)
```





I HAVE WRITTEN INFERENCE AFTER THE 1ST Q SUBPART

Also I have observed for the 1st q r[n] = u[n] - u[n-5] and 3rd q r[n+4] the magnitude plot has the same figure (because for 1st it is 1 for 0 to 5 ,and for 3rd it is 1 for -5 to 0 , flipped), and the phase plot has completely flipped

---

```
END OF Q1
```

Q2. Consider the sinusoid $s[n] = (Acos(\omega_0 n + \phi))$ where A = 2; $\omega_0 = \pi/4$; $\phi = \pi/6$. Compute and plot the DTFT from $[-\pi\ ,\ +\pi]$. Use samples from a finite time window

1. $n = [0, 21]$
2. $n = [0, 201]$

Observe and compare the spectrum in both cases.

Note: While plotting the spectrum please obtain lot of points (eg. 4096) so that the details are not lost.

In [24]:
```
N = 4096
omega = np.pi
w = np.pi / 4
A = 2
phi = np.pi / 6
n = 21
x = np.arange(n + 1)
x
```

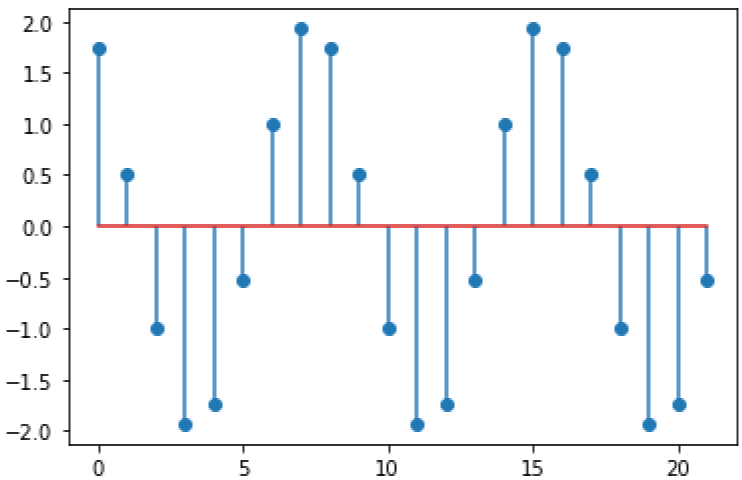
Out[24]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21])

In [25]:
```
s = A * np.cos(w * x + phi)
s
```

Out[25]:
```
array([ 1.73205081,  0.51763809, -1.        , -1.93185165, -1.73205081,
       -0.51763809,  1.        ,  1.93185165,  1.73205081,  0.51763809,
       -1.        , -1.93185165, -1.73205081, -0.51763809,  1.        ,
        1.93185165,  1.73205081,  0.51763809, -1.        , -1.93185165,
       -1.73205081, -0.51763809])
```

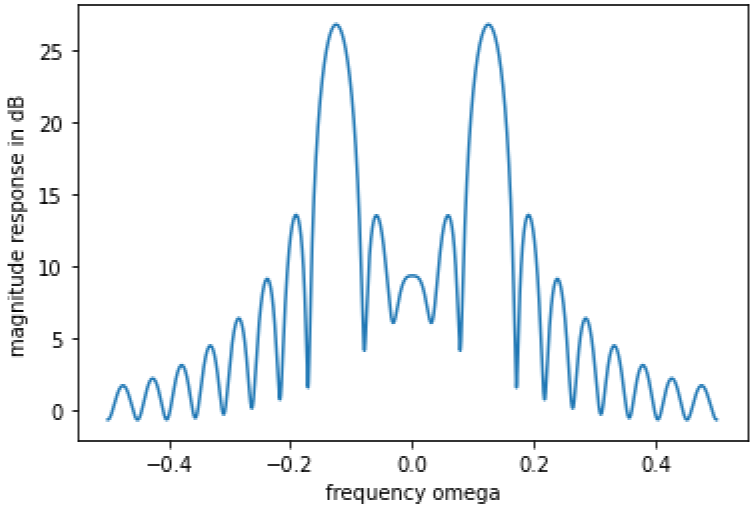
In [26]:
```
plt.stem(x, s)
```

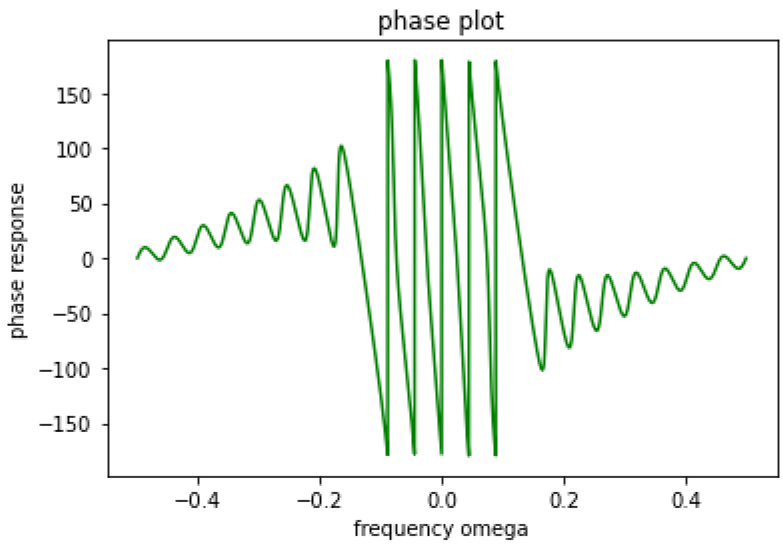
Out[26]: <StemContainer object of 3 artists>



In [27]:
```
def plot(s):
    a = np.ones(1)
    b = s
    c = np.arange(-omega, omega, 2 * omega / N)
    w1, h1 = sig.freqz(b, a, c)
    #     magnitude
    h1db = 20 * np.log10(abs(h1))
    plt.plot(w1 / (2 * np.pi), h1db)
    plt.xlabel('frequency omega')
    plt.ylabel('magnitude response in dB')
    plt.show()

    #     phase
    angles = np.angle(h1, deg=True)
    plt.plot(w1 / (2 * np.pi), angles, 'g')
    plt.title('phase plot')
    plt.xlabel('frequency omega')
    plt.ylabel('phase response')
    plt.show()
```

In [28]:
```
plot(s)
```





In [29]:

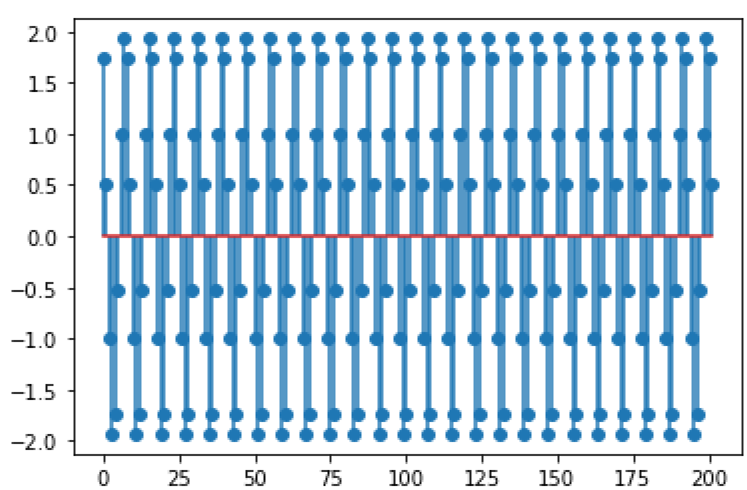
```
n = 201
x = np.arange(n + 1)
s = A * np.cos(w * x + phi)
```

In [30]:

```
plt.stem(x, s)
```

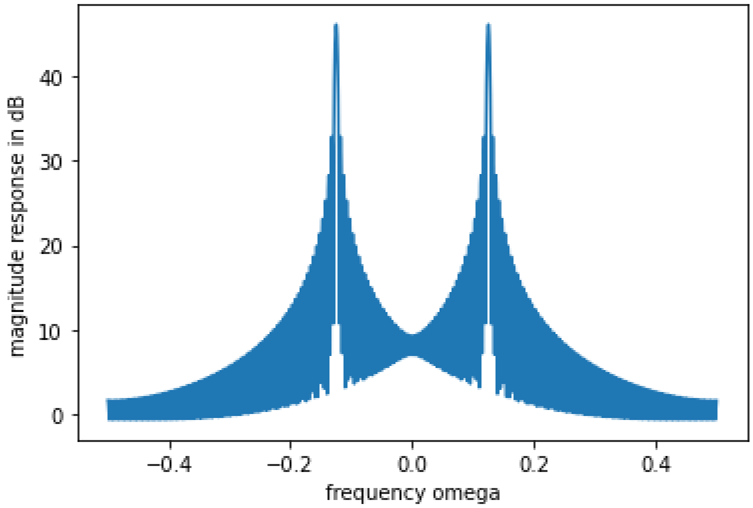
Out[30]:

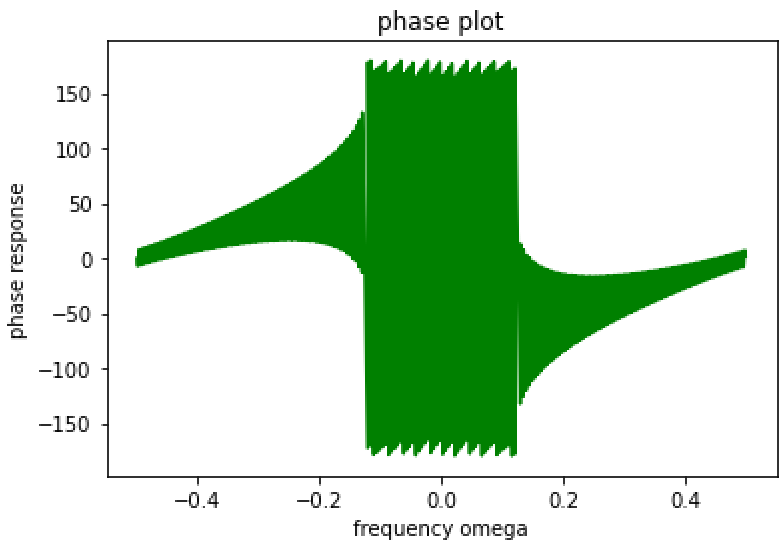
```
<StemContainer object of 3 artists>
```



In [31]:

```
plot(s)
```





Obervation

In the magnitude reponse plot ,there are 2 points in omega for which there is peaking frequency reponse and as the frequency moves towards to +ve infinity and -ve infinity the magnitude reponse is tending to zero (could be a band pass filter).

Similar is the case with phase plot(tends to 0 as both ends of frequency).At the center there is constant switch in the phases maybe from 180 to -180

Also there is a increase in magnitude of frquency reponse(higher value of dB) for case n = [0,201] compared to n =[0,21]

END Q2

Q3. Consider a signal $x[n] \ = u[n] - u[n-6]$. Plot the DTFT of the signal $X(e^{j\omega})$ in $[-\pi \ \ +\pi]$. Consider an expanded version of the signal

$$z[n] = \begin{cases} x[\frac{n}{2}] \ \ , \ \ n \ \ even \\ 0 \ \ , \ \ \ n \ \ odd \end{cases}$$

Plot $Z(e^{j\omega})$(magnitude and phase separately). What is the periodicity of the DTFT? Observe the effect of expanding time axis in the frequency domain.

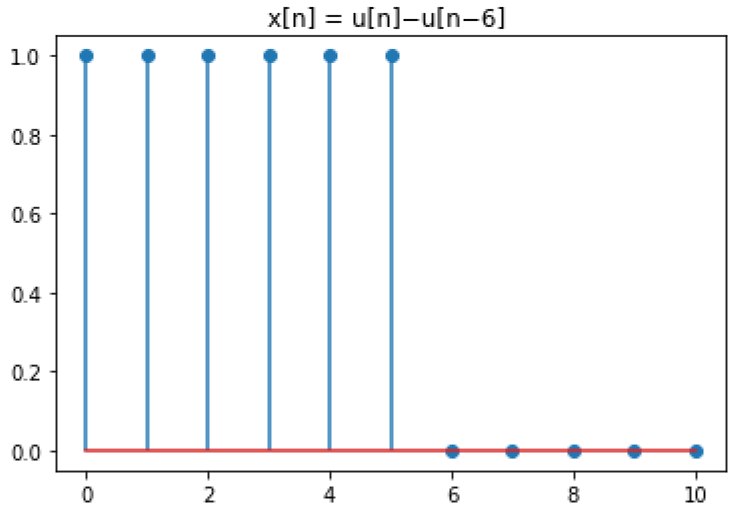
In [32]:

```
N = 4096
omega = np.pi
```

```
n = np.arange(10 + 1)
x = np.zeros(n.size)
x[0:6] = 1

plt.stem(n, x)
plt.title('x[n] = u[n]-u[n-6] ')
```

Out[32]:

```
Text(0.5, 1.0, 'x[n] = u[n]-u[n-6] ')
```

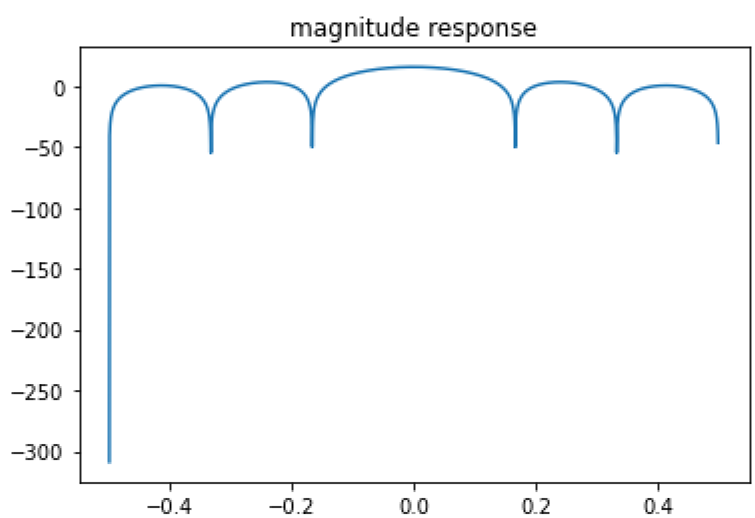


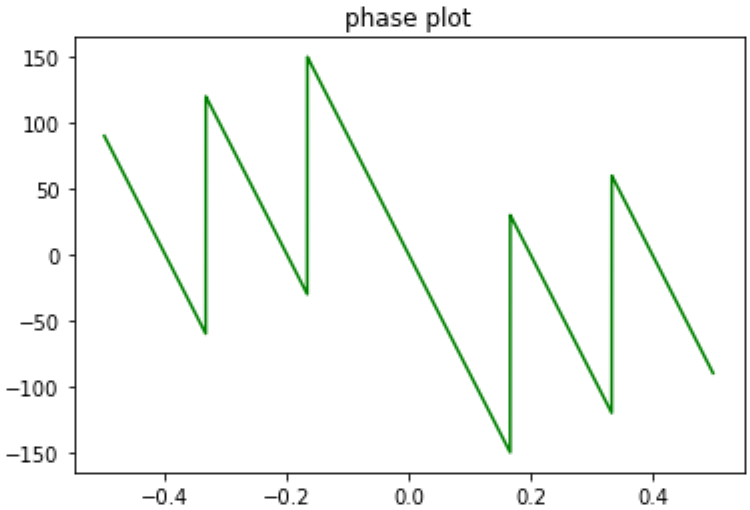
In [33]:

```
def plot(r):
    a = np.ones(1)
    b = r
    c = np.arange(-omega, omega, 2 * omega / N)
    w1, h1 = sig.freqz(b, a, c)
    #      magnitude
    h1db = 20 * np.log10(abs(h1))
    plt.plot(w1 / (2 * np.pi), h1db)
    plt.title('magnitude response')
    plt.show()
    #      phase
    angles = np.angle(h1, deg=True)
    plt.plot(w1 / (2 * np.pi), angles, 'g')
    plt.title('phase plot')
    plt.show()
```

In [34]:

```
plot(x)   #plot for X(exp^jw) both magnitude and phase plot
```





In [35]:

```
z = np.array([0 if i & 1 else x[int(i / 2)] for i in n])
z
```

Out[35]:

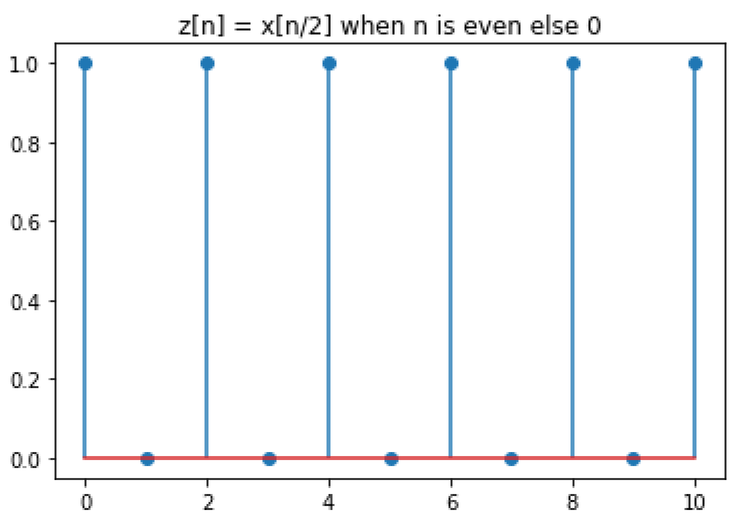
```
array([1., 0., 1., 0., 1., 0., 1., 0., 1., 0., 1.])
```

In [36]:

```
plt.stem(n, z)
plt.title('z[n] = x[n/2] when n is even else 0')
```

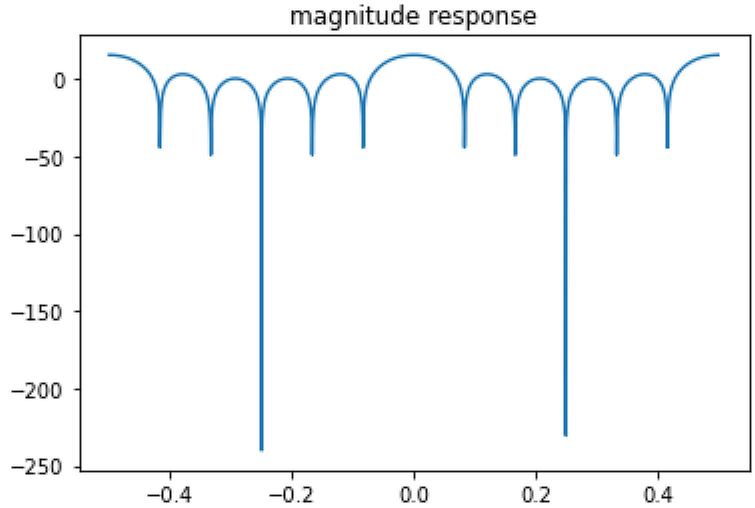
Out[36]:

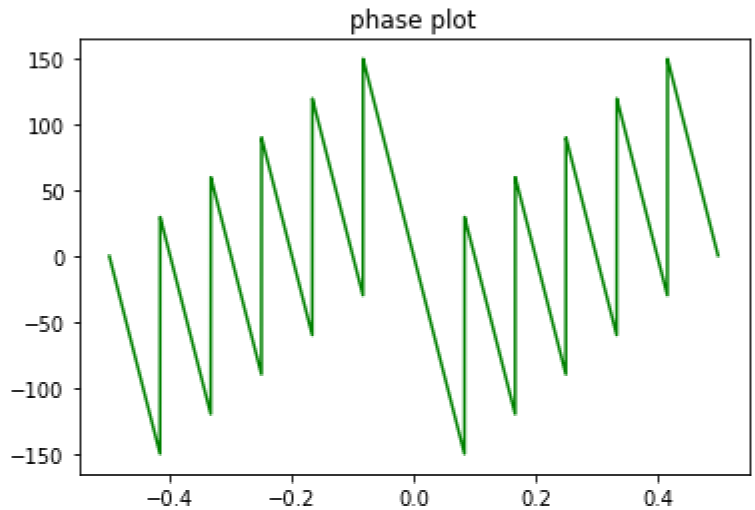
```
Text(0.5, 1.0, 'z[n] = x[n/2] when n is even else 0')
```



In [37]:

```
plot(z)
```





Obervation

Periodicity of DTFT is 2pi

Yes , there is expanding time axis in the frequency domain when compared with x[n] and z[n]

We can see the magnitude response of z[n] has come closer and more dense than x[n] , also can notice that one complete cycle of z[n] is plotted (complete repsonse is seen in that periodic time whereas in x[n] not able to see the complete response in that same time = another down line must be visible)

END Q3

Q4. Consider the signals $x[n] = n(u[n] - u[n-4])$ and $y[n] = 0.9^n(u[n] - u[n-10])$. Find the convolution $z[n] = x[n] * y[n]$ of the signals. Plot $X(e^{j\omega})$, $Y(e^{j\omega})$ and $Z(e^{j\omega})$ (magnitude and phase separately).

In [39]:

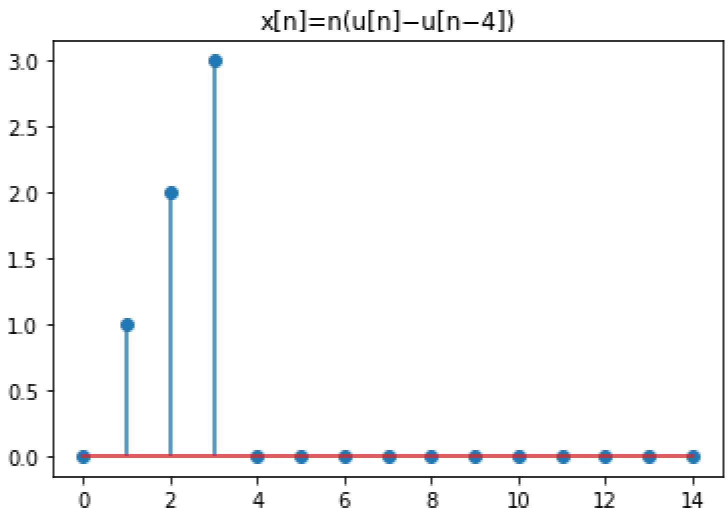
```
N = 4096
omega = np.pi
n = np.arange(15)


def plot(r):
    a = np.ones(1)
    b = r
    c = np.arange(-omega, omega, 2 * omega / N)
    w1, h1 = sig.freqz(b, a, c)
    #      magnitude
    h1db = 20 * np.log10(abs(h1))
    plt.plot(w1 / (2 * np.pi), h1db)
```

```
    plt.xlabel('frequency omega')
    plt.ylabel('magnitude response')
    plt.show()
    #     phase
    angles = np.angle(h1, deg=True)
    plt.plot(w1 / (2 * np.pi), angles, 'g')
    plt.title('phase plot')
    plt.show()
```

In [40]:

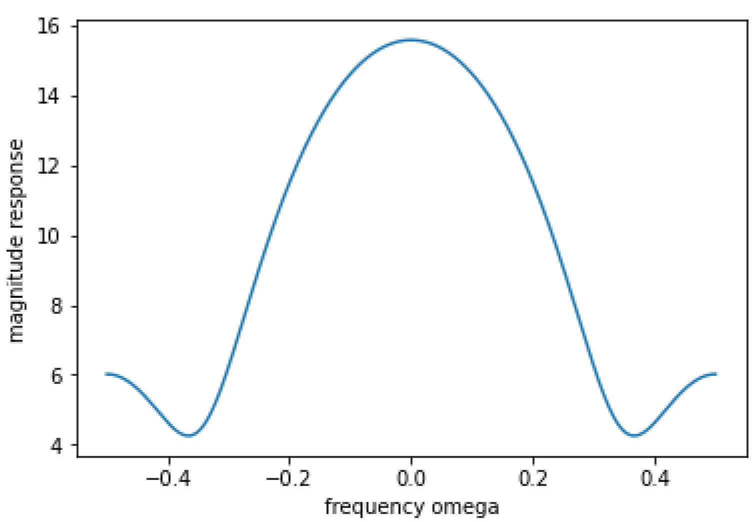
```
x = np.zeros(n.size)
x[0:4] = 1
x = n * x
plt.stem(n, x)
plt.title('x[n]=n(u[n]-u[n-4])')
```

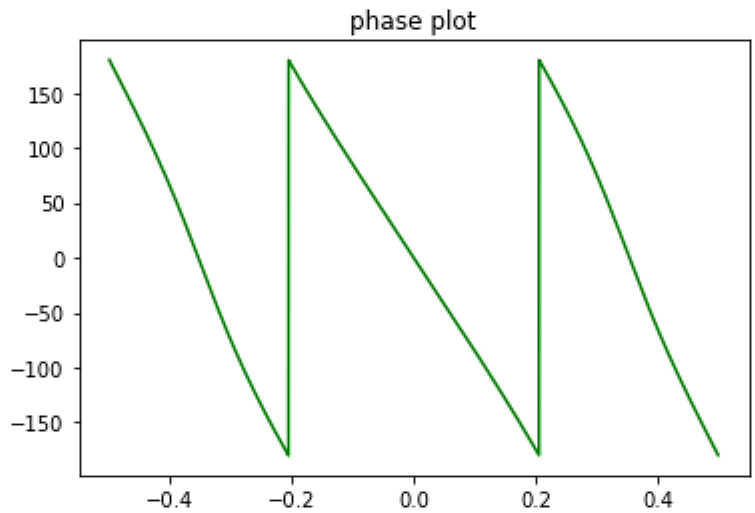
Out[40]: Text(0.5, 1.0, 'x[n]=n(u[n]-u[n-4])')



In [41]:

```
plot(x)   #plot for X(exp^jw) both magnitude and phase plot
```



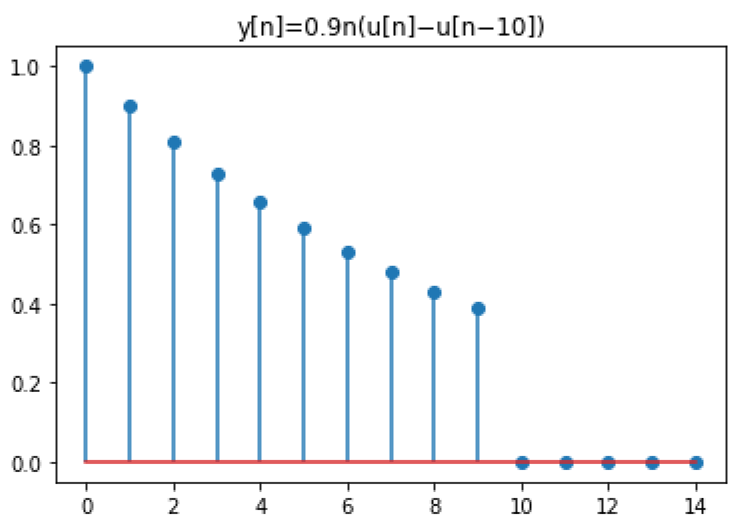


In [42]:

```
y = np.zeros(n.size)
y[0:10] = 1
y = np.power(.9, n) * y
plt.stem(n, y)
plt.title('y[n]=0.9n(u[n]-u[n-10])')
```

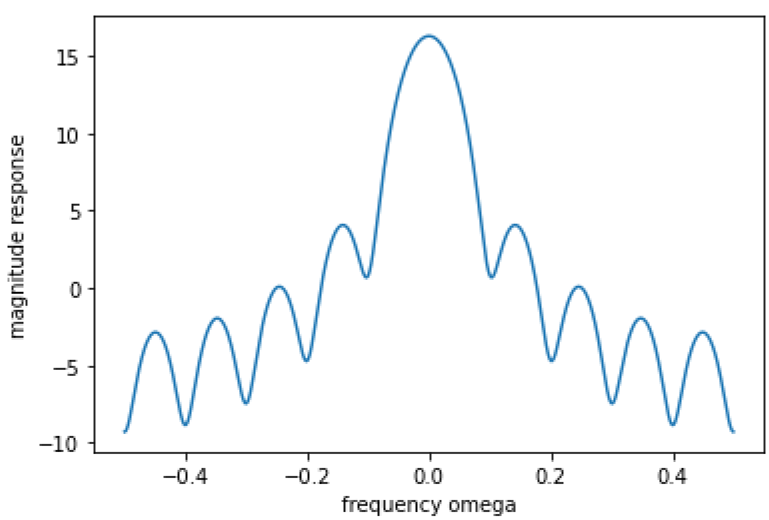
Out[42]:

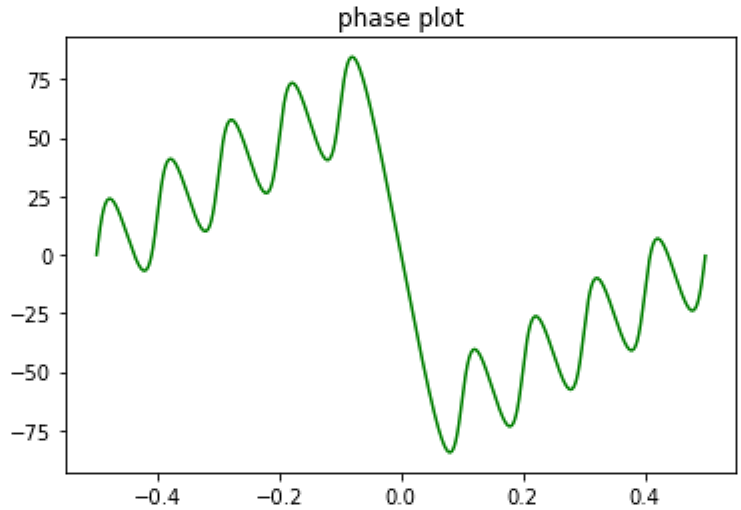
```
Text(0.5, 1.0, 'y[n]=0.9n(u[n]-u[n-10])')
```



In [43]:

```
plot(y)   #plot for Y(exp^jw) both magnitude and phase plot
```



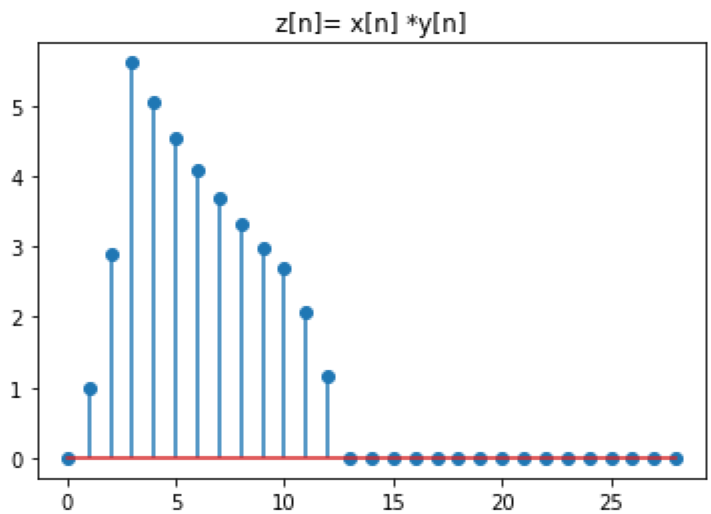


In [44]:

```
z = sig.convolve(x, y)
n2 = np.arange(2 * n[0], 2 * n[-1] + 1)
plt.stem(n2, z)
plt.title('z[n]= x[n] *y[n]')
```

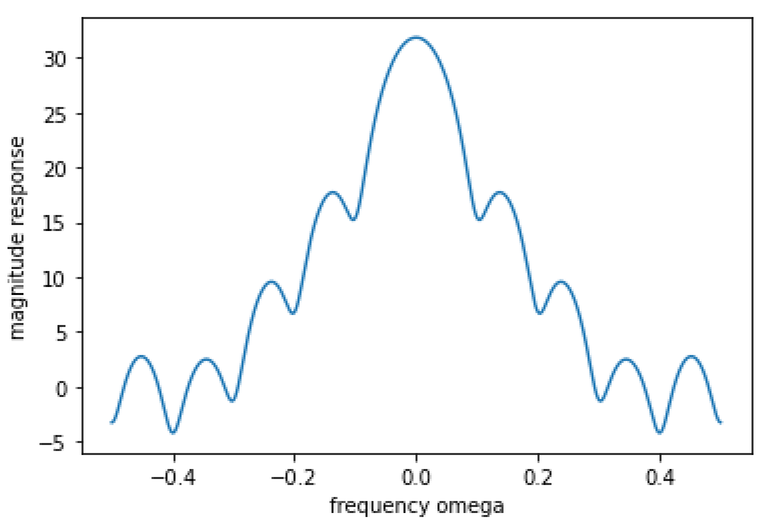
Out[44]:

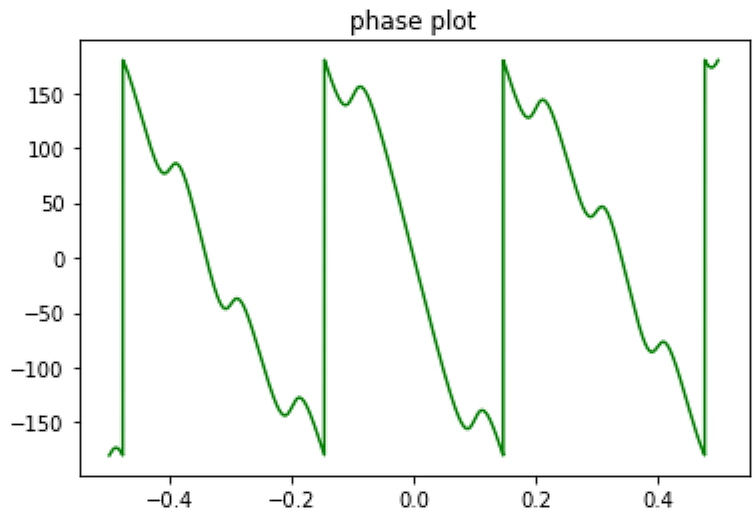
```
Text(0.5, 1.0, 'z[n]= x[n] *y[n]')
```



In [45]:

```
plot(z)
```





Q5. Find the DTFT of the given speech signals that corresponds to two vowels (/a/ and /i/). Which vowel has more high frequency content?

In [46]:
```
import scipy.io.wavfile as wav
```

In [47]:
```
rate_a, data_a = wav.read('a.wav')
rate_a, data_a
```

```
C:\Users\inba2\AppData\Local\Temp/ipykernel_7636/969112589.py:1: WavFileWarning: Chunk (no
n-data) not understood, skipping it.
  rate_a, data_a = wav.read('a.wav')
```

Out[47]:
```
(44100, array([ 39,  47,  24, ..., -48, -48, -29], dtype=int16))
```

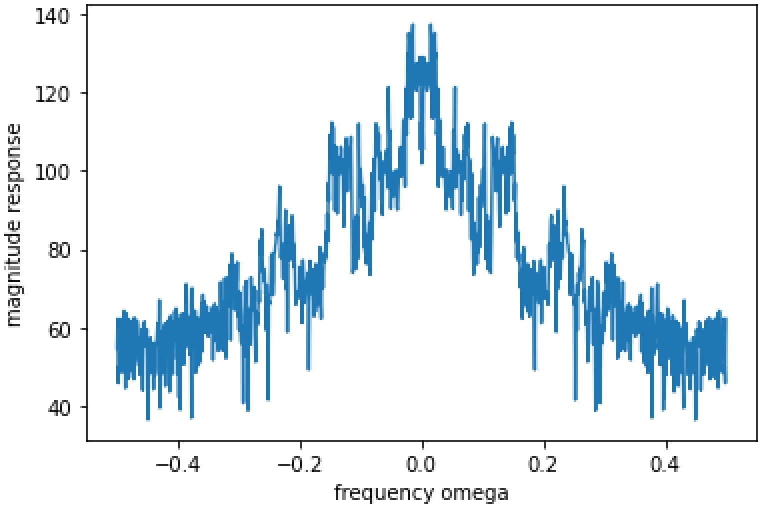
In [48]:
```
rate_i, data_i = wav.read('i.wav')
rate_i, data_i
```

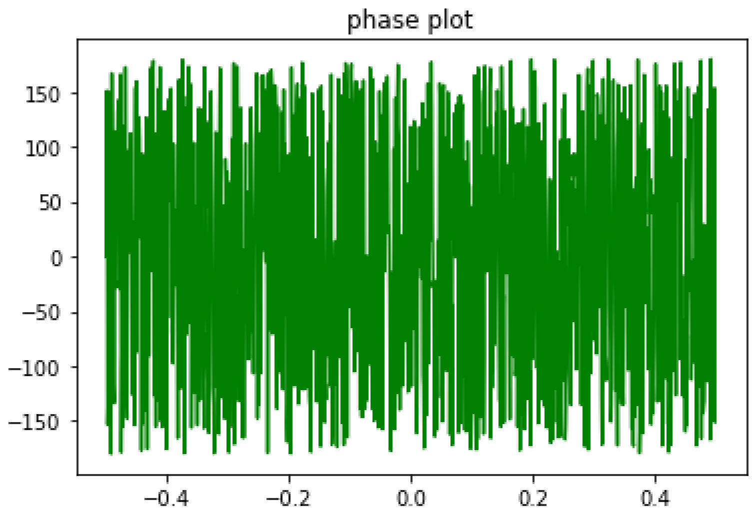
```
C:\Users\inba2\AppData\Local\Temp/ipykernel_7636/1455057468.py:1: WavFileWarning: Chunk (n
on-data) not understood, skipping it.
  rate_i, data_i = wav.read('i.wav')
```

Out[48]:
```
(44100, array([ -2,  16,  42, ..., -16, -22,  -4], dtype=int16))
```

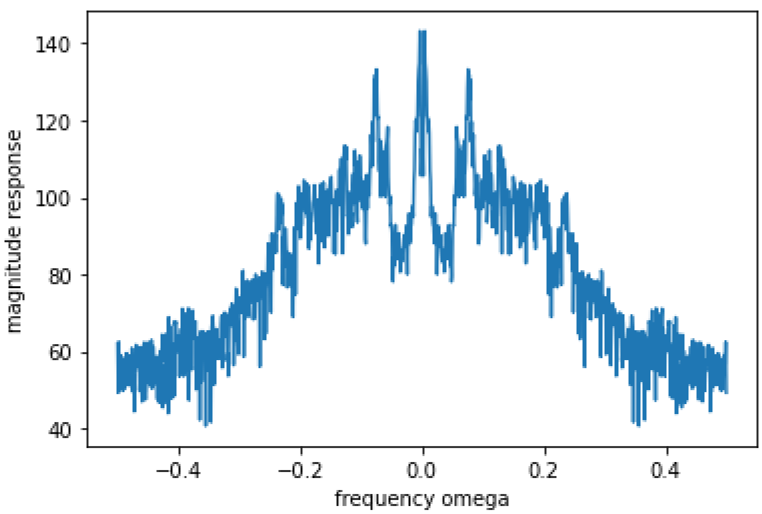
In [49]:
```
N = 1024
omega = np.pi


def plot(r):
    a = np.ones(1)
    b = r
    c = np.arange(-omega, omega, 2 * omega / N)
    w1, h1 = sig.freqz(b, a, c)
#     magnitude
    h1db = 20 * np.log10(abs(h1))
    plt.plot(w1 / (2 * np.pi), h1db)
    plt.xlabel('frequency omega')
    plt.ylabel('magnitude response')
    plt.show()
#     phase
    angles = np.angle(h1, deg=True)
    plt.plot(w1 / (2 * np.pi), angles, 'g')
    plt.title('phase plot')
    plt.show()
```

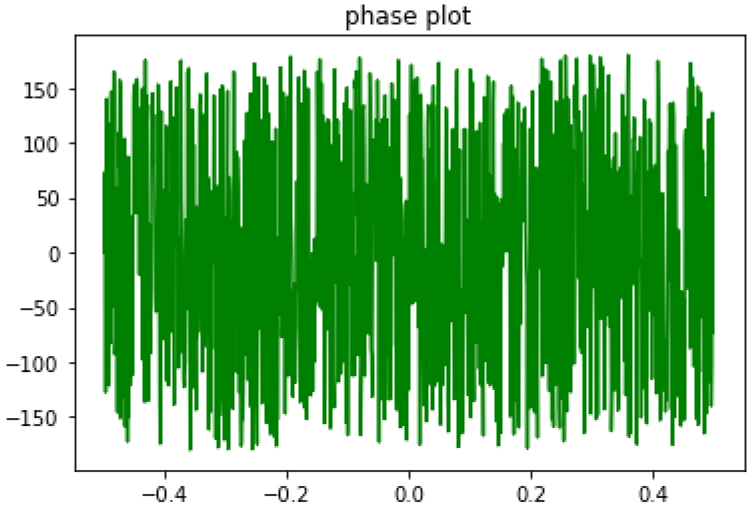
In [50]:
```
plot(data_a)   #a vowel
```





In [51]:
```
plot(data_i)   #i vowel
```

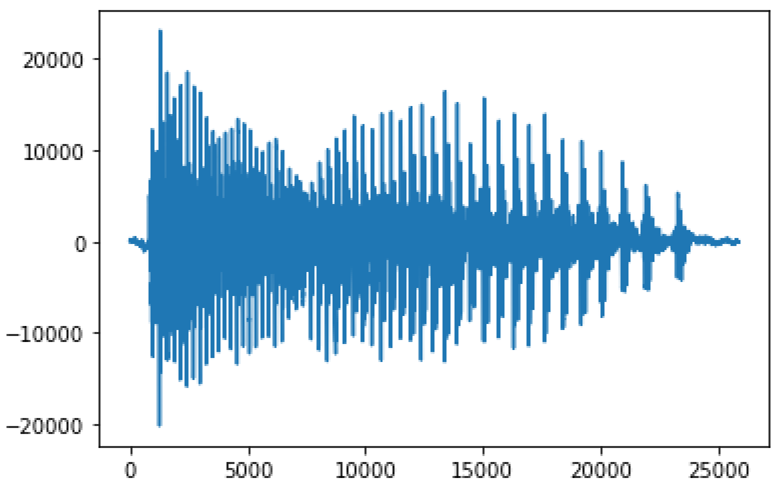


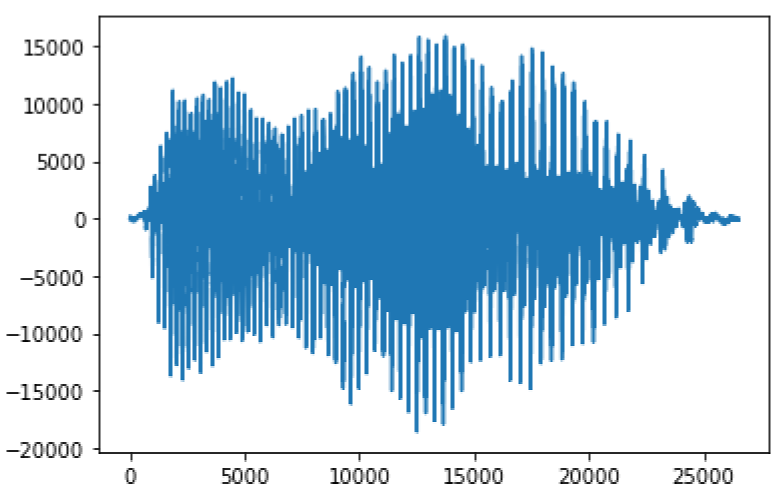


In [57]:

```
freq_count_i = np.sum(abs(np.fft.fft(data_i)))
print("Frequency count in I is: " + str(freq_count_i))
plt.plot(data_a)
plt.show()
plt.plot(data_i)
plt.show()
```

```
Frequency count in I is: 4096878050.085506
```





INFERENCE

From the above Code, we can see that frequency count in I is more than the frequency count in A