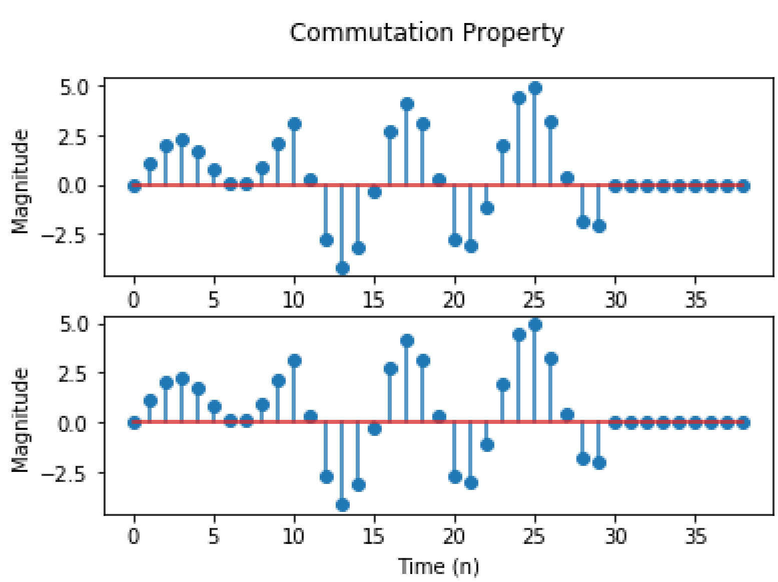
In [29]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import signal
```

Problem 1

In [48]:
```
def u(n):
    return 1 * (n > 0)
```

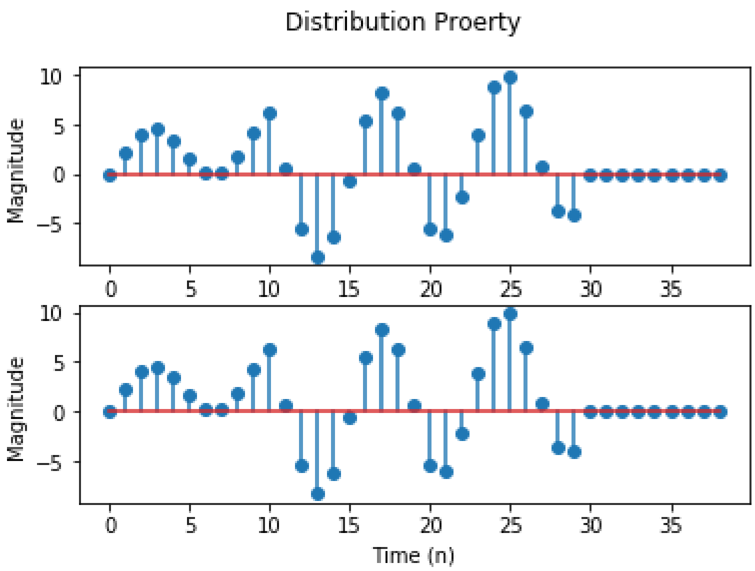
In [60]:
```
n = np.arange(0, 20)

xa = np.cos(np.pi * n / 4)*(u(n + 5) - u(n - 25))
xb = ((0.9)**-n) * (u(n) - u(n - 10))

commutation1 = np.convolve(xa, xb)
commutation2 = np.convolve(xb, xa)

plt.suptitle('Commutation Property')
plt.subplot(2, 1, 1)
plt.stem(commutation1)
plt.xlabel('Time (n)')
plt.ylabel('Magnitude')

plt.subplot(2, 1, 2)
plt.stem(commutation2)
plt.xlabel('Time (n)')
plt.ylabel('Magnitude')
```

Out[60]:
```
Text(0, 0.5, 'Magnitude')
```



In [62]:
```
x1 = xa
x2 = xb
x3 = x2 + x2
left = np.convolve(x1, x3)
right = np.convolve(x1, x2) + np.convolve(x1, x2)
plt.subplot(2, 1, 1)
plt.stem(left)
plt.suptitle('Distribution Proerty')
plt.xlabel('Time (n)')
plt.ylabel('Magnitude')

plt.subplot(2, 1, 2)
plt.stem(right)
plt.xlabel('Time (n)')
plt.ylabel('Magnitude')
```

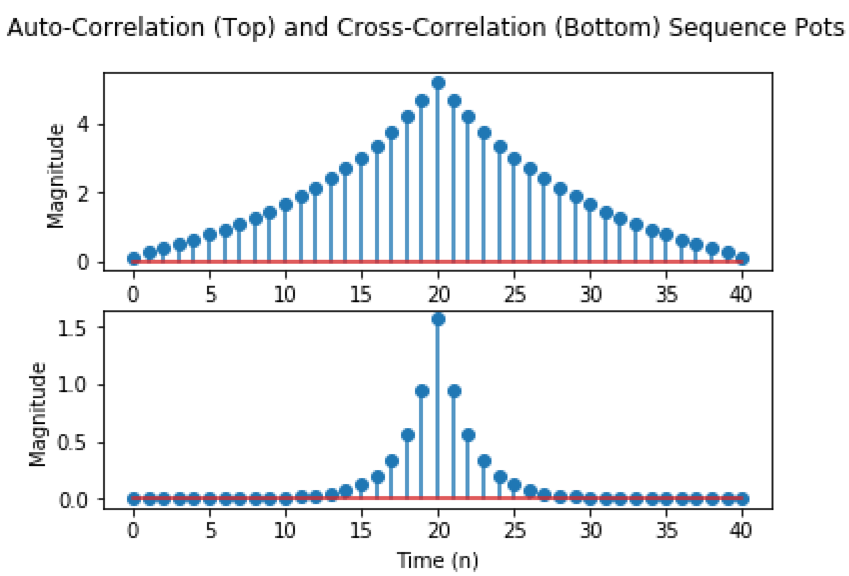
Out[62]: Text(0, 0.5, 'Magnitude')



Problem 2

In [80]:
```
n = np.arange(0, 21, 1)

x = (0.9)**n
y = (0.6)**n

xflip = np.flip(x)
yflip = np.flip(y)

auto = np.convolve(x, xflip)
cross = np.convolve(y, yflip)


plt.suptitle("Auto-Correlation (Top) and Cross-Correlation (Bottom) Sequence P
ots")
plt.subplot(2, 1, 1)
plt.stem(auto)
plt.xlabel('Time (n)')
plt.ylabel('Magnitude')

plt.subplot(2, 1, 2)
plt.stem(cross)
plt.xlabel('Time (n)')
plt.ylabel('Magnitude')
```

Out[80]:
```
Text(0, 0.5, 'Magnitude')
```



Problem 3: Determine the signal parameters Ac, As, r, v0 in terms of the rational function parameters b0, b1, a1 and a2. b0, b1 = numerator coefficient a1, a2 = denominator coefficient x(n)=Ac*(r^n)*cos(piv*0*n)*u(n)* + *As*(r^n)*sin(piv0n)*u(n)

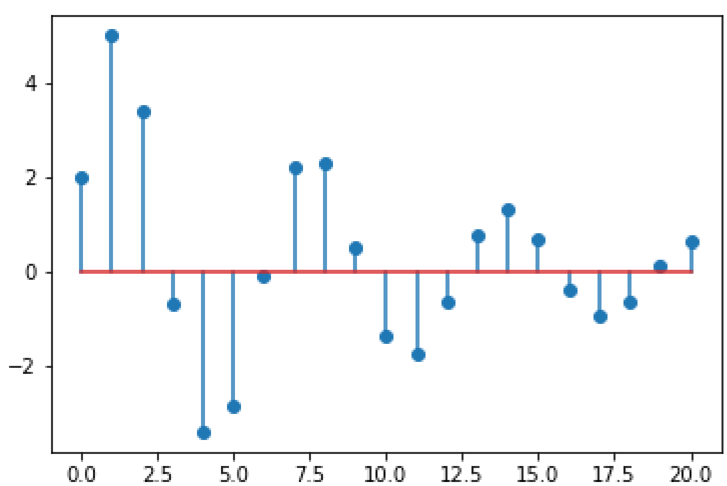
In [13]:
```
def invCCPP(b0, b1, a1, a2):
    Ac = b0
    r = np.sqrt(a2)
    v0 = np.arccos(-a1 / (2*r)) / np.pi
    As = (b1 - a1 * b0 / 2 ) / (r * np.sin(v0 * np.pi))
    return Ac, As, v0, r
```

In [8]:
```
def u(n):
    return 1 * n > 0
```

In [64]:
```
def main():
    b0 = 2
    b1 = 3
    a1 = -1
    a2 = 0.81
    Ac, As, v0, r = invCCPP(b0, b1, a1, a2)
    print(Ac, As, v0, r)
    n = np.arange(0, 21, 1)
    xn = Ac * (r**n) * np.cos(np.pi * v0 * n) + As * (r**n) * np.sin(np.pi * v
0 * n) * u(n)
    # x(n)=Ac*(r^n)*cos(pi*v0*n)*u(n) + As*(r^n)*sin(pi*v0*n)*u(n)
    plt.stem(n, xn)
```

In [12]:
```
if __name__ == '__main__':
    main()
```

```
2 5.3452248382484875 0.31250561891173007 0.9
```



x(n) = 2(0.9^2) *cos(0.3125pi* n) *u(n) + 5.35(0.9)*^2 sin(0.3125pi *n)* u(n) THIS MATCHES WITH THE RESULT OF THE CODE :)

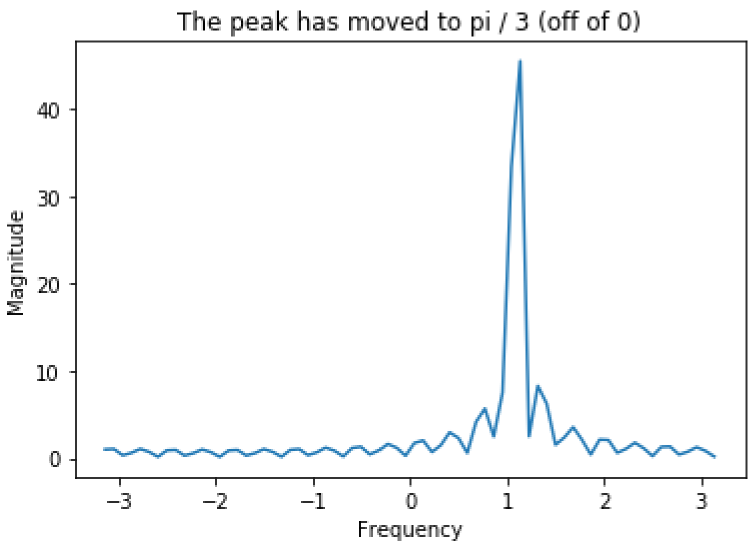
Problem 5

In [14]:
```
from scipy import fftpack
```

In [69]:
```
#function to create a step sequence
def stepseq(n0, n1, n2):
    if((n0 < n1) or (n0 > n2) or (n1 > n2)):
        print("Arguments must satisfy n1 <= n0 <= n2")
    n = np.arange(n1, n2)
    x = np.array((n - n0) > 0)
    x = [int(i) for i in x]
    return x, n

#function to add two signals together
def sig_add(x1, x2, n1, n2):
    n_l = min(n1[0], n2[0])
    n_h = max(n1[-1], n2[-1])
    n = np.arange(n_l, n_h + 1)
    y = np.zeros(len(n), int)
    i = n1[0] - n[0]
    y[i:i + len(x1)] = x1
    i = n2[0] - n[0]
    y[i:i + len(x2)] += x2
    return y, n

# For example, if you want ot create a rect pulse with L = 50
# It can be built by two step functions substracting
# Same as: u[n] - u[n-50].
x1, n1 = stepseq(0, -10, 60)
x2, n2 = stepseq(50, -10, 60)
x2 = [(-i) for i in x2]
x3, n3 = sig_add(x1, x2, n1, n2)
```

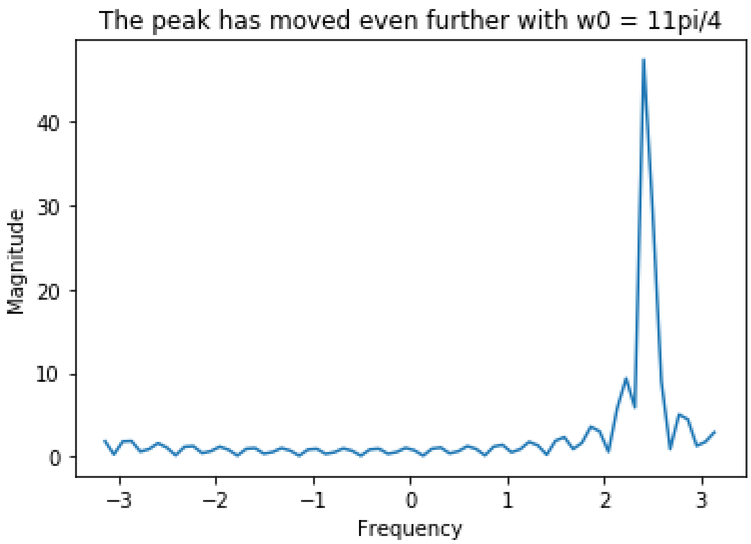
In [86]:
```
w0 = np.pi/3
L = 50

sinusoid = np.exp(1j * w0 * n3)

final = x3 * sinusoid
finalanswer = fftpack.fft(final)
finalanswer = fftpack.fftshift(finalanswer)

plt.plot(np.linspace(-np.pi, np.pi, len(n3)), np.abs(finalanswer))
plt.title('The peak has moved to pi / 3 (off of 0)')
plt.ylabel('Magnitude')
plt.xlabel('Frequency')
```

Out[86]: Text(0.5, 0, 'Frequency')



In [84]:
```
w1 = 11 * np.pi/4
sinusoid2 = np.exp(1j * w1 * n3)

final2 = x3 * sinusoid2
finalanswer2 = fftpack.fft(final2)
finalanswer2 = fftpack.fftshift(finalanswer2)
plt.plot(np.linspace(-np.pi, np.pi, len(n3)), np.abs(finalanswer2))
plt.title('The peak has moved even further with w0 = 11pi/4')
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
```

Out[84]:
```
Text(0, 0.5, 'Magnitude')
```



In [ ]:
```
```