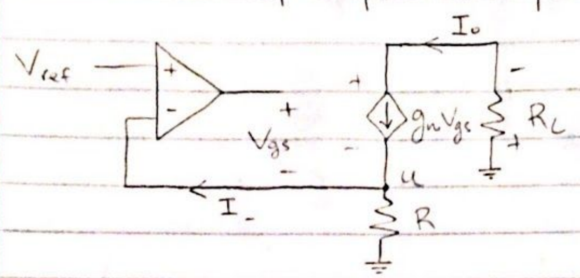
1. Op Amp with output current



a) $g_m V_{gs} = I_o = I_- + I_R$

$$g_m\left(A(V_{ref} - u) - u\right) = I_o$$

$$I_o = I_- + I_R$$
$$I_o = I_- + \frac{u}{R}$$
$$u = R(I_o - I_-)$$

$$g_m\left[A(V_{ref} - I_o R - I_- R) - I_o R + I_- R\right] = I_o$$

$$g_m(AV_{ref} - AI_o R - AI_- R - I_o R + I_- R) = I_o$$

$$g_m A V_{ref} - g_m A I_- R + g_m I_- R = I_o + g_m I_o R A + g_m I_o R$$

$$I_o = \frac{g_m A V_{ref} - g_m A I_- R + g_m I_- R}{1 + g_m R A + g_m R}$$

b) The closest circuit I could find was called "Transresistance Amplifier."

2. Cauchy - Schwarz Inequality

a) $\|\vec{v}\| = \sqrt{(r\cos\theta)^2 + (r\sin\theta)^2} = \underline{r}$

$\|\vec{w}\| = \underline{t}$

b) $\langle \vec{v}, \vec{w} \rangle = \vec{v}^T \vec{w} = \begin{bmatrix} r\cos\theta & r\sin\theta \end{bmatrix} \begin{bmatrix} t\cos\varphi \\ t\sin\varphi \end{bmatrix} = tr\cos\theta\cos\varphi + rt\sin\theta\sin\ldots$

$\langle \vec{v}, \vec{w} \rangle = \boxed{rt\cos(\theta - \phi)}$

c) $\langle \vec{v}, \vec{w} \rangle = \|\vec{v}\| \|\vec{w}\| \cos(\theta - \varphi)$

$\langle \vec{v}, \vec{w} \rangle = \boxed{\|\vec{v}\| \|\vec{w}\| \cos\alpha}$

α is the angle between $\vec{v}$ and $\vec{w}$

We know that $\cos\alpha \leq 1$

So, $\langle \vec{v}, \vec{w} \rangle = \|\vec{v}\| \|\vec{w}\|$ when $\cos\alpha = 1$

otherwise

$\langle \vec{v}, \vec{w} \rangle < \|\vec{v}\| \|\vec{w}\|$ when $\cos\alpha < 1$

Thus,

$$\boxed{\langle \vec{v}, \vec{w} \rangle \leq \|\vec{v}\| \|\vec{w}\|}$$

3. Mechanical Linear Correlation

b)

	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n+3]$	1 2 3 4 0 0 0 0 0 0
k=-3	$\langle \vec{s}_1[n], \vec{s}_2[n+3] \rangle$	8

	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n+2]$	0 1 2 3 4 0 0 0 0 0
k=-2	$\langle \vec{s}_1[n], \vec{s}_2[n+2] \rangle$	6 - 8 = -2

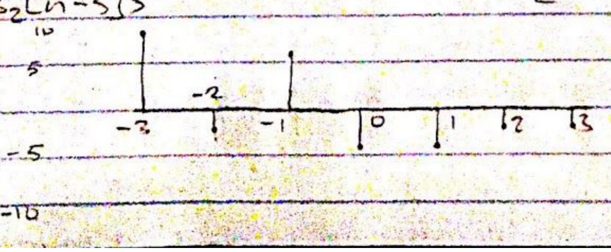
	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n+1]$	0 0 1 2 3 4 0 0 0 0
k=-1	$\langle \vec{s}_1[n], \vec{s}_2[n+1] \rangle$	4 - 6 + 8 = 6

	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n]$	0 0 0 1 2 3 4 0 0 0
k=0	$\langle \vec{s}_1[n], \vec{s}_2[n] \rangle$	2 - 4 + 6 - 8 = -4

	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n-1]$	0 0 0 0 1 2 3 4 0 0
k=1	$\langle \vec{s}_1[n], \vec{s}_2[n-1] \rangle$	-2 + 4 - 6 = -4

	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n-2]$	0 0 0 0 0 1 2 3 4 0
k=2	$\langle \vec{s}_1[n], \vec{s}_2[n-2] \rangle$	2 - 4 = -2

	$\vec{s}_1[n]$	0 0 0 2 -2 2 -2 0 0 0
	$\vec{s}_2[n-3]$	0 0 0 0 0 0 1 2 3 4
k=3	$\langle \vec{s}_1[n], \vec{s}_2[n-3] \rangle$	-2



c) No, they will be different because linear correlation is not commutative. Using IPython notebook gives us:

$$\text{corr}_{\vec{s}_2}(\vec{s}_1)[k] = [-2 \quad -2 \quad -4 \quad -4 \quad -2 \quad 8]$$

The 2 correlations are horizontally flipped.

4. Audio File Matching

a) $\vec{x}_1 = [1\ 1 \ldots 1]^T$

$\vec{x}_2 = [1\ 1 \ldots 1]^T$

$\langle \vec{x}_1, \vec{x}_2 \rangle = \|\vec{x}_1\|^2 = 1+1+\ldots+1$

If $\vec{x}_2 = [1\ -1\ 1\ -1 \ldots 1\ -1]^T$

$\langle \vec{x}_1, \vec{x}_2 \rangle = 1-1+1-1\ldots+1-1 = 0$

To compare similarity between 2 vectors, we take the inner product, the larger it is, more similar they are

b) $\vec{x} = [-1\ 1\ 1\ -1\ 1\ 1\ -1\ 1]^T$

Using IPython gives us,

$corr_{\vec{x}}([1\ 1\ -1]) = [1\ -2\ -1\ 3\ -1\ -1\ 3\ -1\ 0\ 1]$

So, the maximum is at indexes 3 and 6.

If we align
$$\begin{matrix} -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \end{matrix}$$

Then, the maximums at shift $\boxed{k = -1, 2}$

$corr_{\vec{x}}([1\ 1\ 1]^T) = [-1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1])$

The maximums are at shifts $k = -2, -1, 0, 1, 2, 3, 5$

c) I expect to see a peak where [1 2 3] vector matches on both vectors.

When $\vec{x} = [1\ 2\ 3\ 1\ 2\ 2\ 3\ 10]^T$, I expect to see a peak at $k=0$

Using IPython, $corr_{\vec{x}}([1 2 3]) = [3\ 8\ 14\ 11\ 11\ 11\ 15\ 38\ 23\ 10]$

The peak is not where expected it to be. This is because the song vector is <u>not</u> good for correlation. We want a vector that returns the max value when correlated with itself.

5. GPS Receivers

a) I observe that the Gold code peaks at $k=0$.

b) There's no peak in this correlation.

c) There's still no peak in the random noise. This means we can still figure out the satellite in presence of random noise.

d) The same observation as part (c). The correlation is still low and we can find a peak even in real-valued noise.

e) Satellites 4, 7, 13, 19 are present

f) The unknown sequence is [1 -1 -1 -1 1] from Satellite 3

7. Homework Process

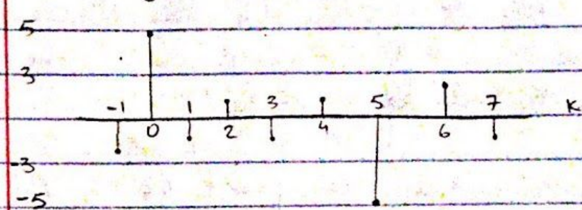
I worked on this homework alone. I read all of the notes, so I was able to complete the homework.

6. Golden Positioning System

a) The satellite is transmitting [1 -1 -1 1]

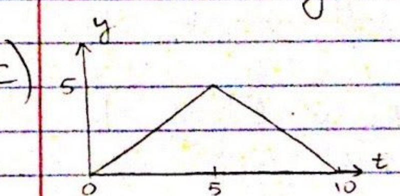
b) Using IPython, we get:

$corr_{\vec{r}}(\vec{g}) = [-1\ 0\ 1\ -2\ 5\ -1\ 1\ -1\ 1\ -5\ 2\ -1\ 0\ 1]$

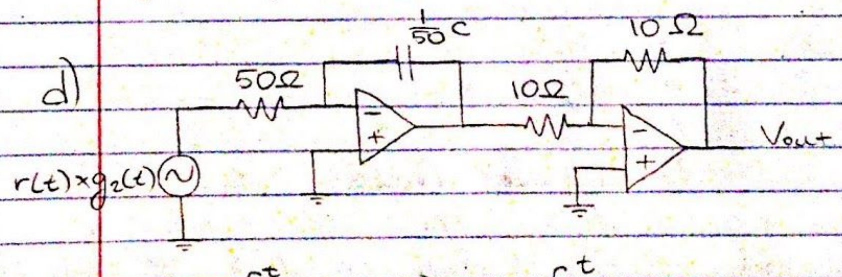


The peak on the plot tell us the similarity of $\vec{r}$ and $\vec{g}$ at that shift value.

c)



d)



$$V_{out} = \int_0^t \frac{r(t) \times g_2(t)}{50\Omega \cdot \frac{1}{50}C}\, dt = \int_0^t r(t) \times g_2(t)\, dt$$

e) $\underline{d_i = c \cdot (T_i - S_i)}$

f) The receiver needs data from $\boxed{4}$ satellites.

EECS16A Homework 12

Question 3: Mechanical Correlation

Part (c)

In [2]:
```
import numpy as np
s1 = [2, -2, 2, -2]
s2 = [1, 2, 3, 4]

# Use the function np.correlate with mode='full' for linear cross corr
print(np.correlate(s1, s2, mode="full"))
print(np.correlate(s2, s1, mode="full"))   # they are flipped horizont
```

```
[ 8 -2  6 -4 -4 -2 -2]
[-2 -2 -4 -4  6 -2  8]
```

Question 4: Audio File Matching

This notebook continues the audio file matching problem. Be sure to have song.wav and clip.wav in the same directory as the notebook.

In this notebook, we will look at the problem of searching for a small audio clip inside a song.

The song "Mandelbrot Set" by Jonathan Coulton is licensed under [CC BY-NC 3.0 (http://creativecommons.org/licenses/by-nc/3.0/)](http://creativecommons.org/licenses/by-nc/3.0/)

If you have trouble playing the audio file in IPython, try opening it in a different browser. I encountered problem with Safari but Chrome works for me.

In [3]:
```
# Part B
import numpy as np
print(np.correlate([-1, 1, 1, -1, 1, 1, -1, 1], [1, 1, -1], mode="full
print(np.correlate([-1, 1, 1, -1, 1, 1, -1, 1], [1, 1, 1], mode="full"
```

```
[ 1 -2 -1  3 -1 -1  3 -1  0  1]
[-1  0  1  1  1  1  1  1  0  1]
```

In [4]:
```
import numpy as np
import wave
import matplotlib.pyplot as plt
import scipy.io.wavfile
import operator
from IPython.display import Audio
%matplotlib inline

given_file = 'song.wav'
target_file = 'clip.wav'
rate_given,  given_signal  = scipy.io.wavfile.read(given_file)
rate_target, target_signal = scipy.io.wavfile.read(target_file)
given_signal  = given_signal[:2000000].astype(float)
target_signal = target_signal.astype(float)
def play_clip(start, end, signal=given_signal):
    scipy.io.wavfile.write('temp.wav', rate_given, signal[start:end].a
    return Audio(url='temp.wav', autoplay=True)

def run_comparison(target_signal, given_signal, idxs=None):
    # Run everything if not called with idxs set to something
    if idxs is None:
        idxs = [i for i in range(len(given_signal)-len(target_signal))
    return idxs, [vector_compare(target_signal, given_signal[i:i+len(t
                  for i in idxs]

play_clip(0, len(given_signal))
```

Out[4]:

-0:45

We will load the song into the variable `given_signal` and load the short clip into the variable `target_signal`. Your job is to finish code that will identify the short clip's location in the song. The clip we are trying to find will play after executing the following block.

In [5]:
```
Audio(url=target_file, autoplay=True)
```

Out[5]:

-0:01

Your task is to define the function 'vector_compare' and run the following code. Because the song has a lot of data, you should use the provided examples from the previous parts of the problem before running the later code. Do you results here make sense given your answers to previous parts of the problem?

In [2]:
```
def vector_compare(short_clip, segment_of_song):
    """This function compares two vectors, returning a number.
    The test vector with the highest return value is regarded as being
    return sum([short_clip[i] * segment_of_song[i] for i in range(len(

print("PART A:")
print(vector_compare(np.array([1,1,1]), np.array([1,1,1])))
print(vector_compare(np.array([1,1,1]), np.array([-1,-1,-1])))
print("PART C:")
print(vector_compare(np.array([1,2,3]), np.array([1,2,3])))
print(vector_compare(np.array([1,2,3]), np.array([2,3,4])))
print(vector_compare(np.array([1,2,3]), np.array([3,4,5])))
print(vector_compare(np.array([1,2,3]), np.array([4,5,6])))
print(vector_compare(np.array([1,2,3]), np.array([5,6,7])))
print(vector_compare(np.array([1,2,3]), np.array([6,7,8])))
```

```
PART A:
3
-3
PART C:
14
20
26
32
38
44
```

Part (e)

Run the following code that runs vector_compare on every subsequence in the song- it will probably take at least 5 minutes. How do you interpret this plot to find where the clip is in the song?

In [ ]:
```
import time

t0 = time.time()
idxs, song_compare = run_comparison(target_signal, given_signal)
t1 = time.time()
plt.plot(idxs, song_compare)
print ("That took %(time).2f minutes to run" % {'time':(t1-t0)/60.0} )
```

Question 5: GPS Receivers

In [6]:
```
%pylab inline
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import sys
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [7]:
```
## RUN THIS FUNCTION BEFORE YOU START THIS PROBLEM
## This function will generate the gold code associated with the satel
## The satellite_ID can be any integer between 1 and 24
def Gold_code_satellite(satellite_ID):
    codelength = 1023
    registerlength = 10

    # Defining the MLS for G1 generator
    register1 = -1*np.ones(registerlength)
    MLS1 = np.zeros(codelength)
    for i in range(codelength):
        MLS1[i] = register1[9]
        modulo = register1[2]*register1[9]
        register1 = np.roll(register1,1)
        register1[0] = modulo

    # Defining the MLS for G2 generator
    register2 = -1*np.ones(registerlength)
    MLS2 = np.zeros(codelength)
    for j in range(codelength):
        MLS2[j] = register2[9]
        modulo = register2[1]*register2[2]*register2[5]*register2[7]*r
        register2 = np.roll(register2,1)
        register2[0] = modulo

    delay = np.array([5,6,7,8,17,18,139,140,141,251,252,254,255,256,25
    G1_out = MLS1;
    shamt = delay[satellite_ID - 1]
    G2_out = np.roll(MLS2,shamt)

    CA_code = G1_out * G2_out

    return CA_code
```

Part (a)

In [8]:
```
def cross_correlation(array1, array2):
    """ This function should return two arrays or a matrix with one ro
    the offset and other to the correlation value. array1 and array2 c
    arrays of equal length.
    Think of array1 as the received signal and array2 as the signature
    The function should return correlation values as well as the indic
    Hint: look up np.correlate
    """

    #correlated_array = #Your code here (it is just one line) np.corre
    correlated_array = np.correlate(array1, array2, 'full')

    #Since both the arrays start at 0, the last "shift" where the sign
    end_index = len(array1)

    #Similarly, the first "shift" where the signals overlap is the neg
    st_index = -len(array2) + 1

    indices = np.arange(st_index, end_index)
    return (indices, correlated_array)
```

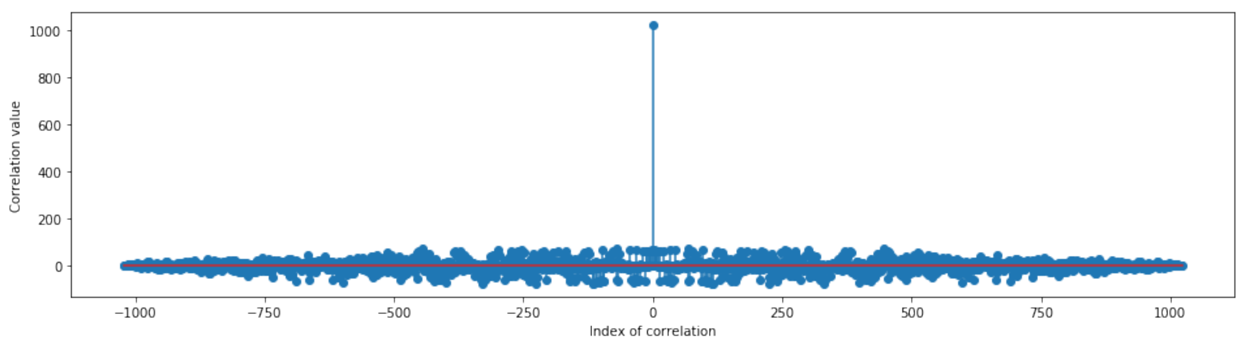
In [9]:
```python
# Plot the auto-correlation of satellite 10 with itself. Fill in the i
array_10 = Gold_code_satellite(10)

(ind_10, self_10) = cross_correlation(array_10, array_10)

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

```
/Users/manlai/miniconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:7: UserWarning: In Matplotlib 3.3 individual lines on a stem pl
ot will be added as a LineCollection instead of individual lines. Thi
s significantly improves the performance of a stem plot. To remove th
is warning and switch to the new behaviour, set the "use_line_collect
ion" keyword argument to True.
  import sys
```

Out[9]: `Text(0, 0.5, 'Correlation value')`



The autocorrelation peaks at 1023 when the signals are perfectly aligned (offset 0). The correlation of a Gold code with a shifted version of itself is not significant.

Part (b)

Plot the cross correlation when array1 = satellite 13 and array2 = satellite10

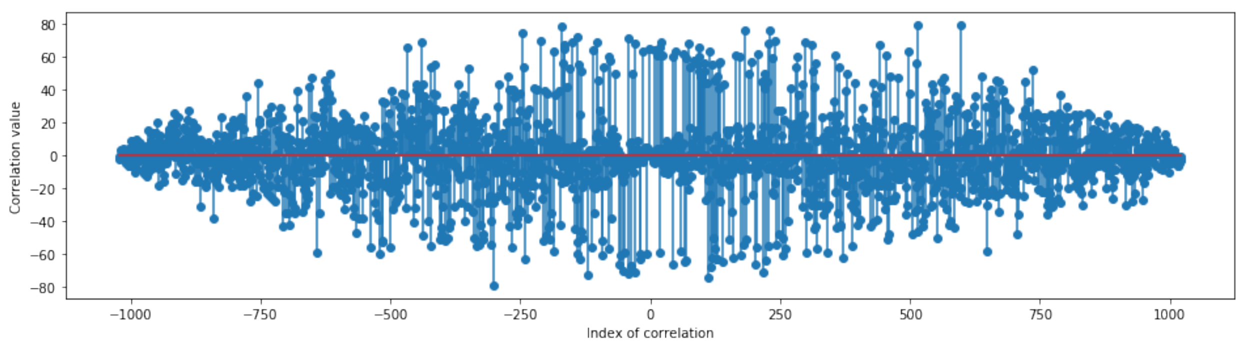
In [10]:
```
array_10 = Gold_code_satellite(10)
array_13 = Gold_code_satellite(13)

(ind_10, self_10) = cross_correlation(array_10, array_13)

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

```
/Users/manlai/miniconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:7: UserWarning: In Matplotlib 3.3 individual lines on a stem pl
ot will be added as a LineCollection instead of individual lines. Thi
s significantly improves the performance of a stem plot. To remove th
is warning and switch to the new behaviour, set the "use_line_collect
ion" keyword argument to True.
  import sys
```

Out[10]: Text(0, 0.5, 'Correlation value')



We see that the cross-correlation of a Gold code of any satellite with any other satellite is very low. This indicates that when given some unknown data, we can differentiate between different satellites.

Part (c)

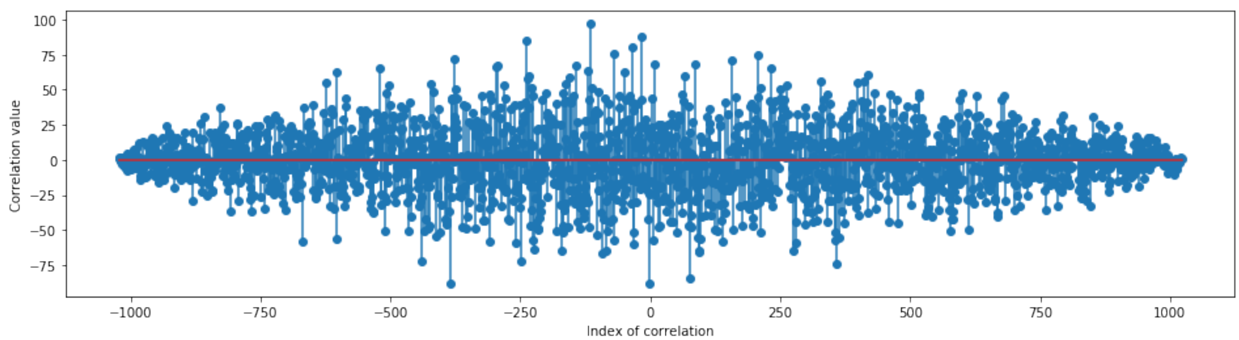
In [11]:
```
## THIS IS A HELPER FUNCTION FOR PART C THAT GENERATES +-1 RANDOM NOIS
def integernoise_generator(length_of_noise):
    noise_array = np.random.randint(2, size = length_of_noise)
    noise_array = 2 * noise_array - np.ones(size(noise_array))
    return noise_array

array_10 = Gold_code_satellite(10)
(ind_10, self_10) = cross_correlation(array_10, integernoise_generator

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

```
/Users/manlai/miniconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:11: UserWarning: In Matplotlib 3.3 individual lines on a stem p
lot will be added as a LineCollection instead of individual lines. Th
is significantly improves the performance of a stem plot. To remove t
his warning and switch to the new behaviour, set the "use_line_collec
tion" keyword argument to True.
  # This is added back by InteractiveShellApp.init_path()
```

Out[11]: `Text(0, 0.5, 'Correlation value')`



We see that the cross-correlation of the Gold code of any satellite with integer noise is very low. This indicates that we can still figure out the presence of a satellite even if it is buried in noise.

Part (d)

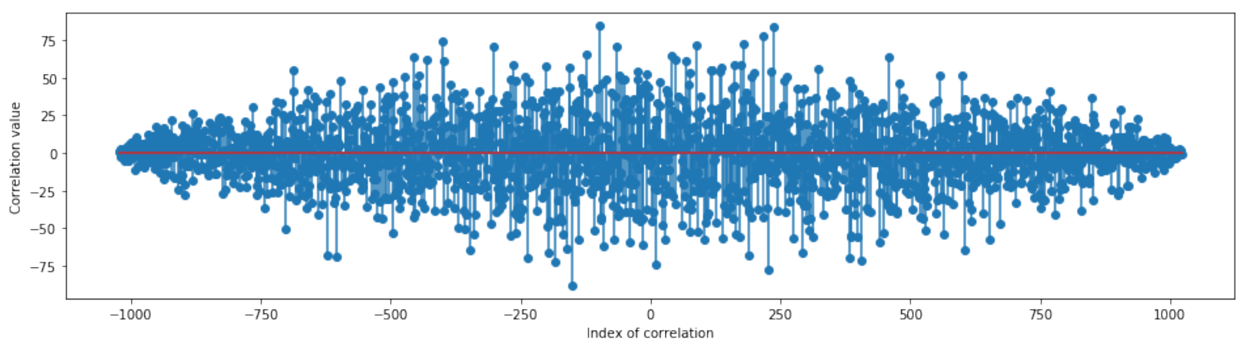
In [12]:
```
## THIS IS A HELPER FUNCTION FOR PART D THAT GENERATES REAL VALUED RAN
def gaussiannoise_generator(length_of_noise):
    noise_array = np.random.normal(0, 1, length_of_noise)
    return noise_array

array_10 = Gold_code_satellite(10)
(ind_10, self_10) = cross_correlation(array_10, gaussiannoise_generato

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

```
/Users/manlai/miniconda3/lib/python3.7/site-packages/ipykernel_launch
er.py:10: UserWarning: In Matplotlib 3.3 individual lines on a stem p
lot will be added as a LineCollection instead of individual lines. Th
is significantly improves the performance of a stem plot. To remove t
his warning and switch to the new behaviour, set the "use_line_collec
tion" keyword argument to True.
  # Remove the CWD from sys.path while we load stuff.
```

Out[12]:
```
Text(0, 0.5, 'Correlation value')
```



We see that the Gold code of any satellite with Gaussian noise is very low. This indicates that we can still figure out the presence of a satellite even if it is buried in Gaussian noise.

Part (e)

Hint: you can use a absolute value threshold of 800 for the cross-correlation to detect if a given satellite is present. np.argwhere may be useful for detecting peak locations.

In [13]:
```
#Now let us see which signals are present in the data signal that is i
signal1 = np.load('data1.npy')
```

In [14]:
```
#Here try plotting the cross-correlations of data1.npy with a few of t
#How can you detect if the satellite is present?
```

In [15]:
```
## This helper function returns 1 if peak (greater than threshold or t
## You do not have to use this function as there are other solutions t

def find_peak(signal, threshold):
    max_value = np.amax(signal)
    min_value = np.amin(signal)
    if max_value > threshold:
        ret_value = 1
    elif min_value < -1 * threshold:
        ret_value = 1
    else:
        ret_value = 0
    return ret_value
```

In [35]:
```
## USE 'np.load' FUNCTION TO LOAD THE DATA
## USE DATA1.NPY AS THE SIGNAL ARRAY
for i in range(1, 25, 1):
    c = cross_correlation(signal1, Gold_code_satellite(i))
    if find_peak(c[1], 800) != 0:
        print("Satellite", i, "is present")
```

```
Satellite 4 is present
Satellite 7 is present
Satellite 13 is present
Satellite 19 is present
```

Part (f)

In [36]:
```
## USE DATA2.NPY AS THE SIGNAL ARRAY
signal2 = np.load('data2.npy')
for i in range(1, 25, 1):
    c = cross_correlation(signal2, Gold_code_satellite(i))
    if find_peak(c[1], 800) != 0:
        print("Satellite", i, "is present")
        print("The data transmitted:", [int(i/abs(i)) for i in c[1] if
```

```
Satellite 3 is present
The data transmitted: [1, -1, -1, -1, 1]
```

Part (g)

In [38]:
```
## USE DATA3.NPY AS THE SIGNAL ARRAY

signal3 = np.load('data3.npy')
for i in range(1, 25, 1):
    c = cross_correlation(signal3, Gold_code_satellite(i))
    if find_peak(c[1], 800) != 0:
        print("Satellite", i, "is present")
```

```
Satellite 5 is present
Satellite 20 is present
```

In [49]:
```
## We know that the data is 1, 1, -1, -1, -1, so we just find the posi
## plot the appropriate cross_correlation and find the location of the
## Do this for as many satellites as there are present
for i in [5, 20]:
    data = list(Gold_code_satellite(i))
    data = np.append(data, data)
    data = np.append(data, -3 * data)
    c = cross_correlation(data, signal3)
    print("Delay of satellite", i, "is", [j for j in range(len(c[1]))
```

```
Delay of satellite 5 is 1528
Delay of satellite 20 is 1022
```

In [1]:
```
# PROBLEM 6

# Part B
import numpy as np
print(np.correlate([1, 1, -1, 1, -1, -1, -1, 1, -1, 1], [1, 1, -1, 1,
```

```
[-1  0  1 -2  5 -1  1 -1  1 -5  2 -1  0  1]
```

In [ ]: