In [262]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from sklearn import linear_model
import scipy.stats as st
from statsmodels.tsa.stattools import adfuller
```

Downloading and cleaning data

This data was taken from Berkeley Earth. The link to the data is:

http://berkeleyearth.lbl.gov/auto/Regional/TMAX/Text/global-land-TMAX-Trend.txt

I copy and pasted the relevant parts of this page into a txt file, and then extracted it using the python numpy package. Berkeley Earth takes temperature samplings from multiple weather stations across the globe. These temperatures are given in terms of anomalies, which means the difference between the mean of the data and each observations value. Below I quickly use this mean data to convert each data point back to the raw celsius temperature.

In [3]:

```
# Data downloaded from the following source:
# http://berkeleyearth.lbl.gov/auto/Regional/TMAX/Text/global-land-TMAX-Trend.txt

data = np.loadtxt('/Users/danielbarkhorn/Desktop/SCU_2016-2017/Spring/MATH_123/temperature_data.txt')[:,0:4
]
monthly_estimates = [8.12, 9.01, 11.35, 14.43, 17.37, 19.42, 20.26, 19.83, 18.07, 15.12, 11.72, 9.08]
```

In [226]:

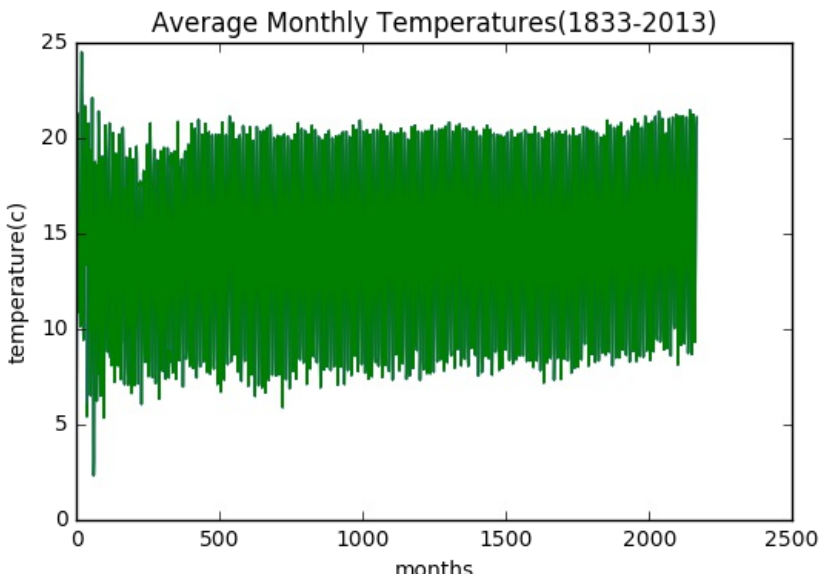
```
# Temperature comes in form 'anomaly'. Here I am making them all celcius data
for i in range(data.shape[0]):
    data[i][2] += monthly_estimates[int(data[i][1])-1]
data = np.nan_to_num(data)
```

Graphing the data

First we will graph the data to get a visual understanding of it. Then we will go into performing our statistical tests, that will help us further define our data.

In [10]:

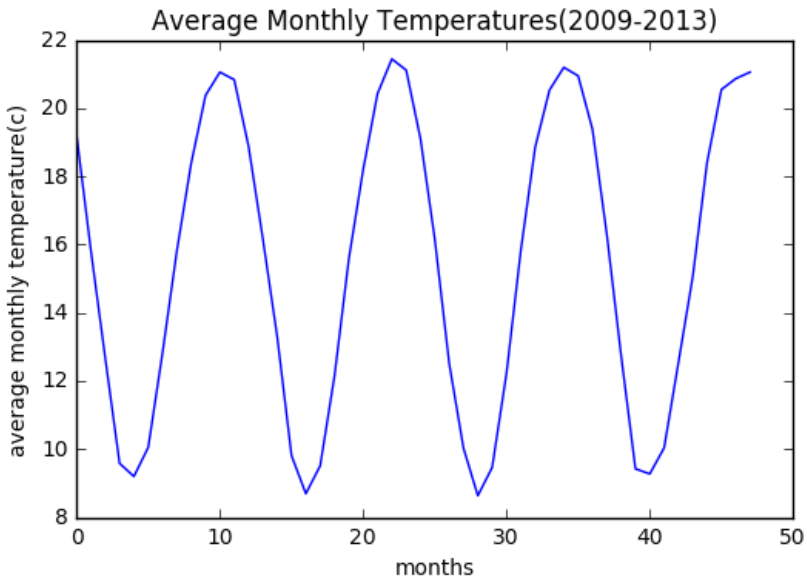
```
plt.plot(data[:,2])
plt.xlabel('months')
plt.ylabel('temperature(c)')
plt.title('Average Monthly Temperatures(1833-2013)')
plt.show()
```



Data is hard to visualize, let's look at the most recent couple of years

In [14]:

```
plt.plot(data[:,2][-48:])
plt.xlabel('months')
plt.ylabel('average monthly temperature(c)')
plt.title('Average Monthly Temperatures(2009-2013)')
plt.show()
```



There is clearly a pattern occurring here-- as is expected with monthly temperature data

Innvestigating Autocorrelation

When it comes to time series data, it is often useful to look at the data with various *lags*. When you lag time series data you look at datapoints that are not adjacent, but as far apart as the lag. For example, our data we would expect that a lag of 12 would yield significant results, for comparing each month to the month the previous year would be perhaps more relevant than comparing it to the month right before it.

To measure the effectiveness of a specific lag, we can use the autocorrelation coefficient. The autocorrelation coefficient r_k tells us the correlation of the data with itself given lag k.

$$r_k = \frac{\sum_{i=k+1}^{n}(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{i=1}^{n}(Y_t - \bar{Y})^2}$$

In [201]:

```
# Compute autocorrelation on given data with lag = k
def autoCorr(data, k):
    n = len(data); num = 0; denom = 0;
    Y_bar = np.nansum(data)/n
    for i in range(k+1, n):
        temp = (data[i]-Y_bar)*(data[i-k]-Y_bar)
        num += temp if not np.isnan(temp) else 0
    for i in range(n):
        temp = (data[i]-Y_bar)**2
        denom += temp if not np.isnan(temp) else 0
    return num/denom
```

It has been shown that for a time series made of randomly generated numbers (white noise) the autocorrelation coefficients have a distribution that is approximated by a normal curve that has $\mu = 0$ and $\sigma = \frac{1}{\sqrt{n}}$ where n is the number of observations in the series.

We can leverage this fact to determine whether or not our data is white noise by using a simple hypothesis test.

$$H_0 : data\ is\ white\ noise$$

$$H_A : data\ is\ not\ white\ noise$$

For a white noise time series, we can use the Z-Chart to determine with a certain percent accuraccy what range all of our autocorrelations should lie within. For example, if we wanted to know with 99 accuracy whether or not our data was white noise we could see if all our autocorrelations were within the range $\pm \frac{2.575}{\sqrt{n}}$.

In [269]:

```
# Range
alpha = 2.575/(n)**.5
print('alpha:', alpha)
```

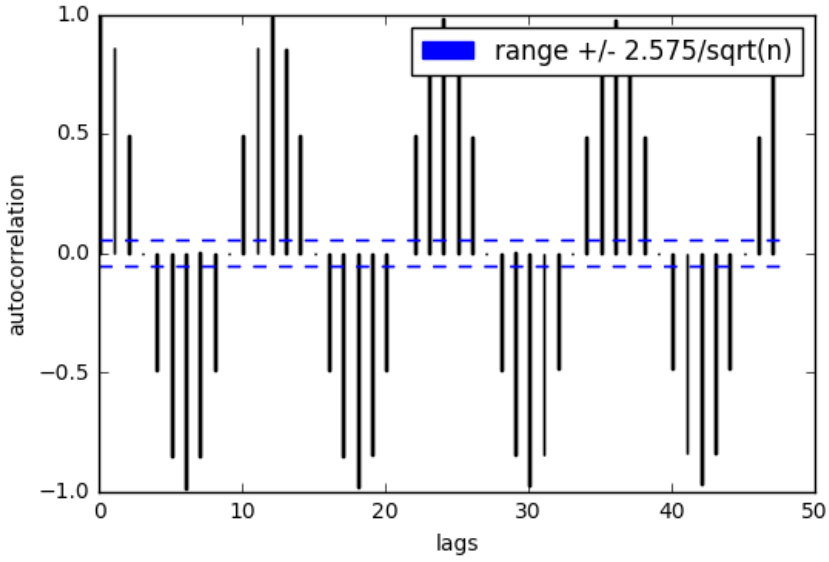
```
alpha: 0.05530286037189806
```

In [267]:

```
# Computing the first 48 autocorrelations
autoCorrs = []
for i in range(48):
    autoCorrs.append(autoCorr(data[:,2], i))
n = len(data[:,2])
```

In [263]:

```
# Graphing the first 48 autocorrelations and the range +/- 2.575/sqrt(n)
plt.bar(range(48), autoCorrs, width = .1)
plt.plot((0,48),(alpha,alpha), 'b--')
plt.plot((0,48),(-1*alpha,-1*alpha), 'b--')
plt.xlabel('lags')
plt.ylabel('autocorrelation')
blue_patch = mpatches.Patch(color='blue', label='range +/- 2.575/sqrt(n)')
plt.legend(handles=[blue_patch])
plt.show()
```



As we can clearly see, this data is *not* white noise-- for all of these values lie outside the range,like we suspected. We see this because many of our lags are not within the range specified. There we *reject the null hypothesis and accept the alternative hypothesis*.

In [214]:

```
# Finding p-value
lag_12 = autoCorrs[12]
p_z = lag_12 * (n)**.5
p = 1 - st.norm.cdf(p_z)
print(p)
```

```
0.0
```

Because our data is so highly correlated, we get a p-value that is basically zero. I use the highest lag (12), because in order for us to believe it is white noise everything must be below the range. The scipy packages norm function probably will not return digits to such a

degree, which leads to our p-vale being 0. This just shows that we are *very* sure this data is not white noise. This is a statistically significant p-value.

From this graph we also see that there may be a pattern, or a relationship between our lags. This would make sense considering the fact that our data is monthly. Next we will investigate seasonality by using the partial autocorrelation coefficient.

Partial Autocorrelation Coefficient

In order to investigate specific lags of timeseries data, we must consider them unrelated to other lags. For example, if we know that Y_t and Y_{t-1} are correlated we also know that Y_{t-1} and Y_{t-2} are correlated. Therefore to investigate the correlation between Y_t and Y_{t-2} we must remove the effect of Y_{t-1} in between.

The partial autocorrelation does this. The partial autocorrelation coefficient is found by creating a regression of Y_t against $Y_{t-1}, \ldots, Y_{t-k}$.

$$Y_t = b_0 + b_1 Y_{t-1} + b_2 Y_{t-2} + \ldots + b_k Y Y_{t-k}$$

Then the partial autocorrelation for each lag, is the corresponding coefficient in the regression. We can use this information to conduct another hypothesis test. We will use the same range as before, but now each coefficient from our regession will tell us whether or not our data is seasonal with respect to that lag.

$$H_0 : data\ has\ no\ seasonality$$

$$H_A : data\ has\ some\ seasonality$$

In [264]:

```
# choosing to look at the first 48 lags (4 years)
# compiling the data to train our regression model on.
pac_data = []
for i in range(48,n):
    pac_data.append(data[i-48:i,2].tolist())

# breaking up compiled data into training and testing sets.
# note the training set will be first 47 numbers the test will be the last number from pac_data rows.
pac_data_train = [i[0:-1] for i in pac_data]
pac_data_train = np.nan_to_num(pac_data_train)
pac_data_test = [i[-1] for i in pac_data]
pac_data_test = np.nan_to_num(pac_data_test)
```

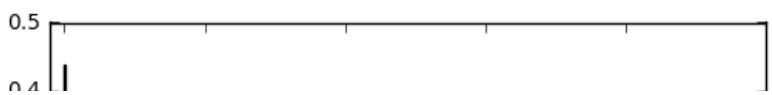
In [265]:

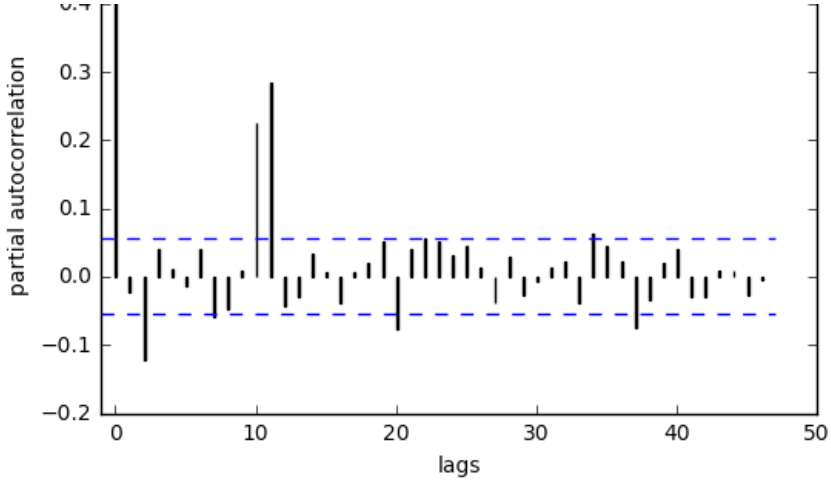
```
# Creating regression using sklearn package
pac_regr = linear_model.LinearRegression()
pac_regr.fit(pac_data_train, pac_data_test)
# reversing the array
coefs = pac_regr.coef_[: :-1]
```

Now that we have our fit regression and its coefficients we can observe the lags. Here we can use the same range as before, to see whether we have any significant lags. Significant lags imply *seasonality*, which is something we would expect seeing how our data is monthly.

In [200]:

```
# Graphing the first 48 partial autocorrelations and the range +/- 2.575/sqrt(n)
plt.bar(range(47), coefs, width = .1)
plt.plot((-1,47),(alpha,alpha), 'b--')
plt.plot((-1,47),(-1*alpha,-1*alpha), 'b--')
plt.xlabel('lags')
plt.ylabel('partial autocorrelation')
plt.xlim([-1,50])
plt.show()
```





Here we observe that there is seasonality to our data. We get significant lags at one month ago, 11 months ago, and 12 months ago. Notice however that unlike before the lag corresponding to 24 months ago is not very high. This is because partial autocorrelation disregards the intermediate lags.

In [270]:

```
# What is the value of our 12th lag? (note zero indexing)
coef_12 = coefs[11]
print('partial autocorrelation coefficient 12:', coef_12)
```

```
partial autocorrelation coefficient 12: 0.331392085926
```

From this value compared to alpha and this graph, we can with 99% accurracy *reject the null hypothesis and accept the alternative hypothesis*.

We can also find the p-value to see how sure we are that the data is seasonal.

In [222]:

```
# Finding p-value for 12th lag
coef_12 = coefs[11]
p_z = coef_12 * (n)**.5
p = 1 - st.norm.cdf(p_z)
print(p)
```

```
0.0
```

Again, our p-value is 0. This is becasue it is *very* clear that our data is seasonal. In other words not only are we 99% sure we are essentially 100% sure that we have seasonality. This is absolutely statistically significant.

Stationarity

For time series data, stationarity means that data has no upward or downward trend. For example, for our data if we found that there is an upward trend, we could say that in the last ~200 years the average temperature on earth has been rising.

We can test for stationarity by using *unit root tests*. One of which is the Augmented Dickey-Fuller Test. Dickey Fuller utilizes the following regression.

$$Y_t^{'} = \phi Y_{t-1} + b_1 Y_{t-1}^{'} + b_2 Y_{t-2}^{'} + \ldots + b_p Y_{t-p}^{'}$$

$$Y_t^{'} = Y_t - Y_{t-1}$$

Using the Augmented Dickey Fuller test, we look at the following statistic.

$$DF_t = \frac{\hat{\phi}}{SE(\hat{\phi})}$$

Then this statistic is compared to a table given by Dickey Fuller. Given the number of samples, we can guess with a % certainty whether or not our data is stationary.

$$H_0 : data\ is\ nonstationary$$

$$H_A : data\ is\ stationary$$

To check these hypotheses, we look at the p-value of our given statistic using table (web.sgh.waw.pl/~mrubas/EP/TabliceStatystyczneDF.doc). On the table we look at model 2 with n > 500. Form here we can see that in order to know with 1% certainty whether or not our data is stationary, we can compare our DF_t statistic to the value 3.433. I use a statsmodel python package to help compute the Augmented Dickey Fuller value.

In [233]:

```
adf = adfuller(data[:,2])
print('ADF Statistic: %f' % adf[0])
print('p-value: %f' % adf[1])
print('Critical Values', adf[4]['1%'])
```

```
ADF Statistic: -5.223909
p-value: 0.000008
Critical Values -3.43340799047
```

Due to our low p-value and critical value, we can reject the null hypothesis. We believe this data is stationary. Again, this is a statistically significant finding.

I for one am unconvinced, I am going to look at only the last 50 years worth of data and consider the stationarity of that data.

In [237]:

```
# use 600 because the last 600 months corresponds to the last 50 years
recent_Data = data[-600:]
recent_adf = adfuller(recent_Data[:,2])
print('ADF Statistic: %f' % recent_adf[0])
print('p-value: %f' % recent_adf[1])
print('Critical Values', recent_adf[4]['5%'])
```

```
ADF Statistic: -2.172167
p-value: 0.216502
Critical Values -2.86649325574
```

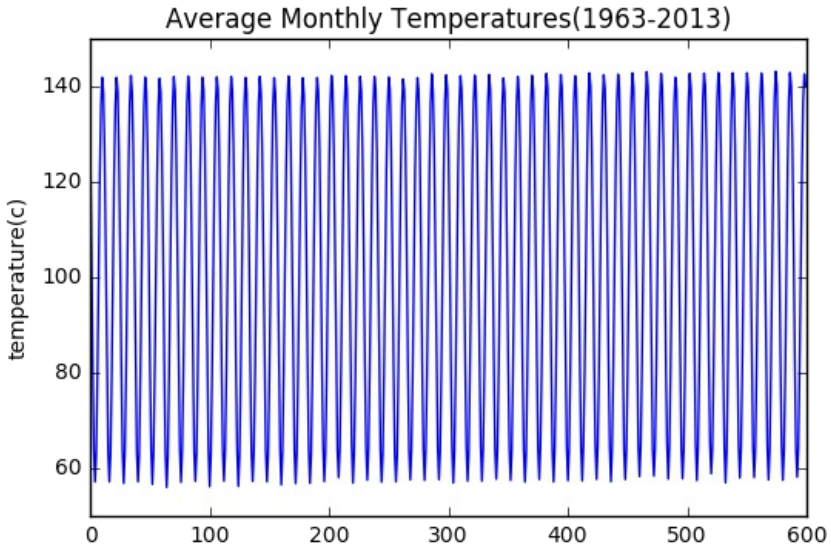
Here our p-value suggests that we can only reject the null hypothesis with 21% accuracy. Therefore, there is a 21% chance that our trend is nonstationary. In other words, this data and the Augmented Dickey Fuller test implies that there is a 21% chance the global temperature has been rising in the past 50 years. Here, we do not have statistically significant findings-- because of our high p-value.

More Graphing & Seasonal/First Differencing

To further this analysis, I will continue graphing. I will first graph the last 50 years worth of data alone, and then I will graph the seasonally differenced data. We found earlier that our data had a seasonality with lag 12, so I will be graphing $Y_t^{'} = Y_t - Y_{t-12}$. Lastly, we saw there was also a significant lag at 1. So I will also do first differencing after the seasonal differenecing. This will result in the following : $Y_t^{''} = Y_t^{'} - Y_{t-1}^{'}$

In [238]:

```
plt.plot(data[-600:,2])
plt.xlabel('months')
plt.ylabel('temperature(c)')
plt.title('Average Monthly Temperatures(1963-2013)')
plt.show()
```



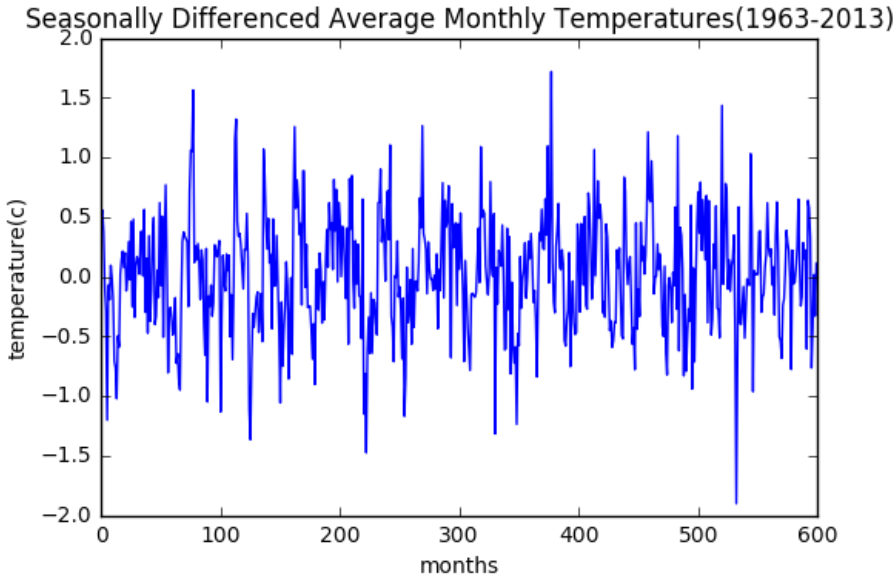
months

In [241]:

```
sd_data = [data[i,2] - data[i-12,2] for i in range(-600,0)]
```

In [243]:

```
plt.plot(sd_data)
plt.xlabel('months')
plt.ylabel('temperature(c)')
plt.title('Seasonally Differenced Average Monthly Temperatures(1963-2013)')
plt.show()
```

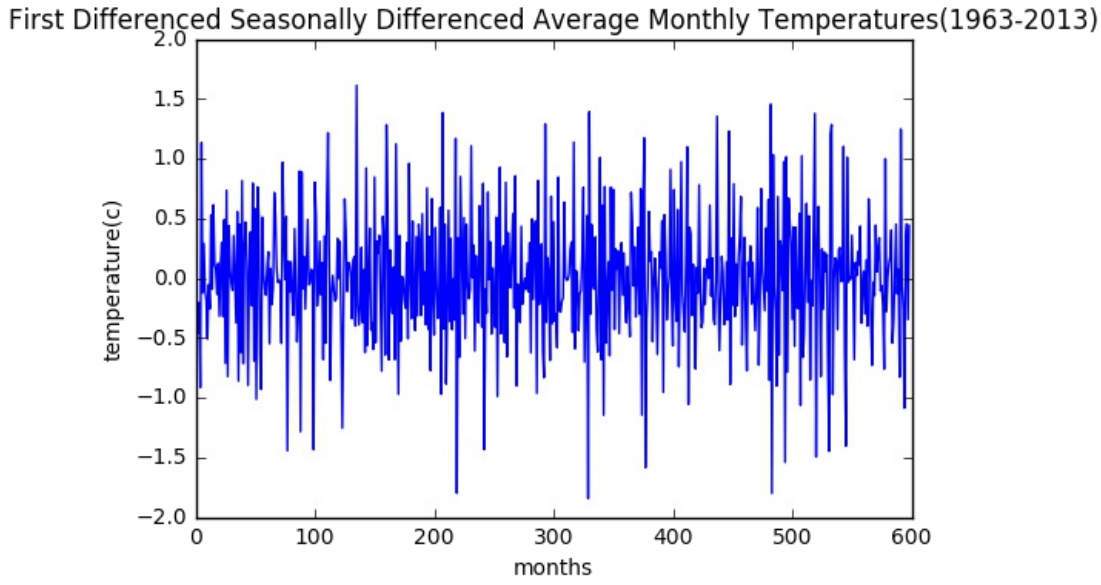


In [245]:

```
sdfd_data = [sd_data[i]-sd_data[i-1] for i in range(-599,0)]
```

In [246]:

```
plt.plot(sdfd_data)
plt.xlabel('months')
plt.ylabel('temperature(c)')
plt.title('First Differenced Seasonally Differenced Average Monthly Temperatures(1963-2013)')
plt.show()
```



From these graphings, it is difficult to observe nonstationarity. This supports the findings of our previous hypothesis testing.

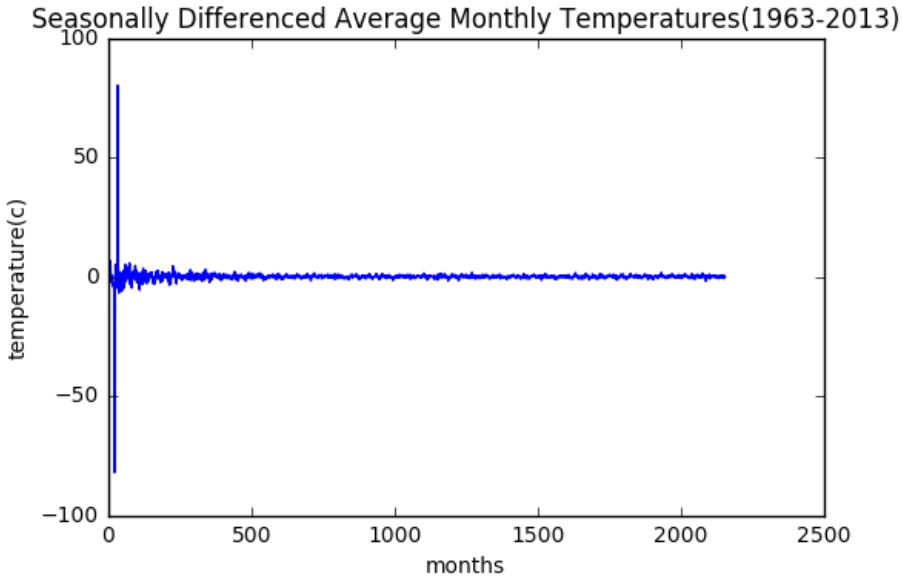
Differenced graphing considering all data

In [252]:

```
sd_data_all = [data[i,2] - data[i-12,2] for i in range(-n+12,0)]
```

In [253]:

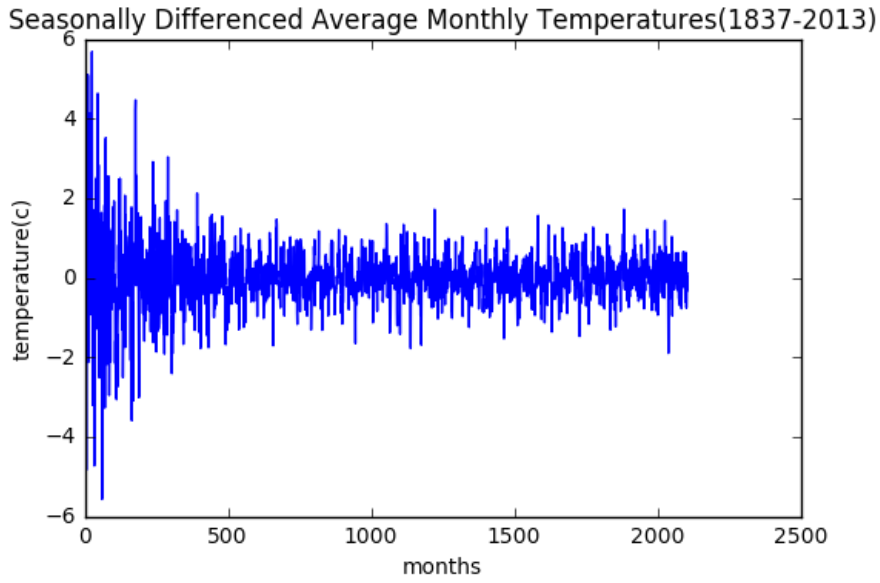
```
plt.plot(sd_data_all)
plt.xlabel('months')
plt.ylabel('temperature(c)')
plt.title('Seasonally Differenced Average Monthly Temperatures(1833-2013)')
plt.show()
```



Obvious outlier at beginning.

In [258]:

```
# Graphing with outlier removed
plt.plot(sd_data_all[50:])
plt.xlabel('months')
plt.ylabel('temperature(c)')
plt.title('Seasonally Differenced Average Monthly Temperatures(1837-2013)')
plt.show()
```



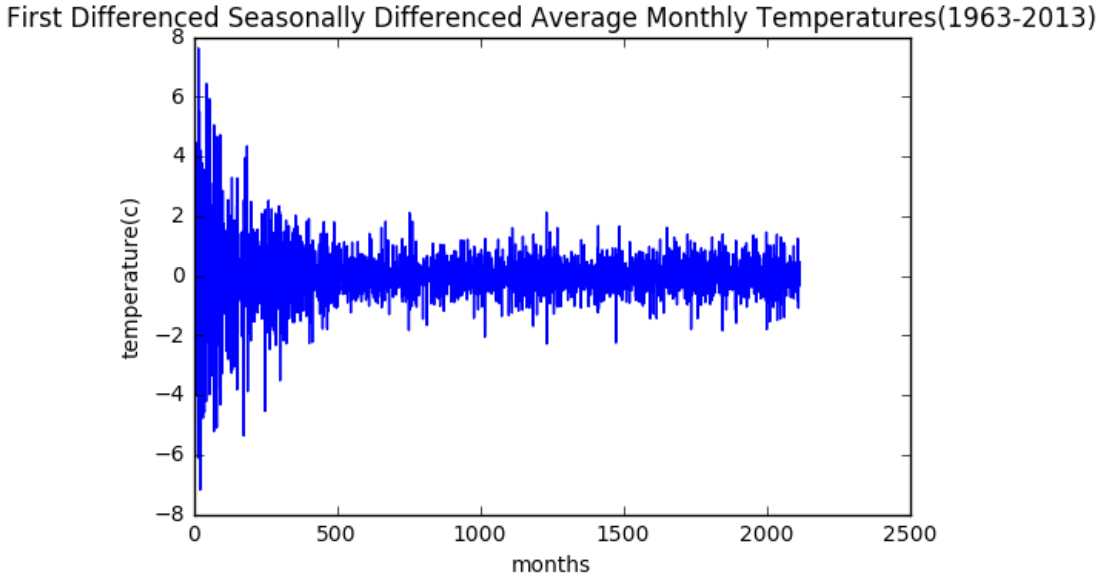
In [259]:

```
sdfd_data_all = [sd_data_all[i]-sd_data_all[i-1] for i in range(-n+51,0)]
```

In [260]:

```
plt.plot(sdfd_data_all)
plt.xlabel('months')
plt.ylabel('temperature(c)')
```

```
plt.title('First Differenced Seasonally Differenced Average Monthly Temperatures(1837-2013)')
plt.show()
```



Conclusion

In this report I tested this data in a number of ways. First I performed basic tests to determine whether the data was random and to determine which lags were significant. Both of these tests leveraged the idea that a randomly created timeseries has an autocorrelation distribution that has a normal distribution with $\mu = 0$ and $\sigma = \frac{1}{\sqrt{n}}$. Both of these tests I found the results that agreed with my intuition given the data.

Next I tested the stationarity of the data. For this I used the Augmented Dickey Fuller test. I found that according to this statistic, the data as a whole had over a 99% cance of being stationary. On the other hand, the last 50 years worth of data had only a 79% chance of being stationary.