LAB-5
ARIMA MODEL

AIM: To implement ARIMA Model.
DESCRIPTION:

Time series is a collection of data points collected at constant time intervals. These are analyzed to determine the long term trend so as to forecast the future or perform some other form of analysis.There are 2 things that differentiates time series from regular regression problem,

- It is time dependent.
- Along with an increasing or decreasing trend, most TS have some form of seasonality trends, i.e. variations specific to a particular time frame.

Checking stationarity in Time series:

A TS is said to be stationary if its statistical properties such as mean, variance remain constant over time. Most of the TS models work on the assumption that the TS is stationary. Intuitively, we can sat that if a TS has a particular behaviour over time, there is a very high probability that it will follow the same in the future. Also, the theories related to stationary series are more mature and easier to implement as compared to non-stationary series.

Stationarity is defined using very strict criterion. However, for practical purposes we can assume the series to be stationary if it has constant statistical properties over time, ie. the following:

- Constant mean
- Constant variance
- Autocovariance that does not depend on time

We can check stationarity using the following:

1.**Plotting Rolling Statistics**:We can plot the moving average or moving variance and see if it varies with time. By moving average/variance at any instant ‘t’, we’ll take the average/variance of the last year, i.e. last 12 months. But again this is more of a visual technique.

2.**Dickey-Fuller Test**:This is one of the statistical tests for checking stationarity. Here the null hypothesis is that the TS is non-stationary. The test results comprise of a Test Statistic and some Critical Values for difference confidence levels. If the ‘Test Statistic’ is less than the ‘Critical Value’, we can reject the null hypothesis and say that the series is stationary.

Making Time series as stationary:

There are 2 major reasons behind non-stationaruty of a TS:

- **Trend**:varying mean over time
- **Seasonality**: variations at specific time-frames.

The underlying principle is to model or estimate the trend and seasonality in the series and remove those from the series to get a stationary series. Then statistical forecasting techniques can be implemented on this series. The final step would be to convert the forecasted values into the original scale by applying trend and seasonality constraints back.

Estimating & Eliminating Trend:

Some techniques are used to estimate or model this trend and then remove it from the series. Most commonly used techniques are:

- **Aggregation:** taking average for a time period
- **Smoothing:**taking rolling averages
- **PolynomialFitting:** fit a regression model

Smoothing techniques:

Moving Average:

In this approach, we take average of 'k' consecutive values depending on the frequency of time series.

A drawback in this particular approach is that the time-period has to be strictly defined.In this case we can take yearly averages but in complex situations like forecasting a stock price, its difficult to come up with a number. So we take a 'weighted moving average' where more recent values are given a higher weight. There can be many technique for assigning weights. A popular one is exponentially weighted moving average where weights are assigned to all the previous values with a decay factor.

Eliminating Trend and Seasonality

The simple trend reduction techniques discussed before don't work in all cases, particularly the ones with high seasonality. Lets discuss two ways of removing trend and seasonality:

1.Differencing: Taking the difference with a particular time lag.One of the most common methods of dealing with both trend and seasonality is differencing. In this technique, we take the difference of the observation at a particular instant with that at the previous instant. This mostly works well in improving stationarity.

2.**Decomposition:M**odeling both trend and seasonality and removing them from the model.In this approach, both trend and seasonality are modeled separately and the remaining part of the series is returned.

Forecasting a Time Series:

ARIMA stands for Auto-Regressive Integrated Moving Averages. The ARIMA forecasting for a stationary time series is nothing but a linear (like a linear regression) equation. The predictors depend on the parameters (p,d,q) of the ARIMA model:

1.**Number of AR (Auto-Regressive) terms (p):**AR terms are just lags of dependent variable. For instance if p is 5, the predictors for x(t) will be x(t-1)….x(t-5).

2.**Number of MA (Moving Average) terms (q)**:MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for x(t) will be e(t-1)….e(t-5) where e(i) is the difference between the moving average at ith instant and actual value.

3.**Number of Differences (d):** These are the number of nonseasonal differences, i.e. in this case we took the first order difference. So either we can pass that variable and put d=0 or pass the original variable and put d=1. Both will generate same results.

An importance concern here is how to determine the value of ‘p’ and ‘q’. We use two plots to determine these numbers. Lets discuss them first.

1.**Autocorrelation Function (ACF):**It is a measure of the correlation between the the TS with a lagged version of itself. For instance at lag 5, ACF would compare series at time instant ‘t1’…’t2’ with series at instant ‘t1-5’…’t2-5’ (t1-5 and t2 being end points).

2.**Partial Autocorrelation Function (PACF)**:This measures the correlation between the TS with a lagged version of itself but after eliminating the variations already explained by the intervening comparisons. Eg at lag 5, it will check the correlation but remove the effects already explained by lags 1 to 4.

In this plot, the two dotted lines on either sides of 0 are the confidence interevals. These can be used to determine the ‘p’ and ‘q’ values as:

1.**p**– The lag value where the PACF chart crosses the upper confidence interval for the first time. If you notice closely, in this case p=2.

2.**q**– The lag value where the ACF chart crosses the upper confidence interval for the first time. If you notice closely, in this case q=2.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize']=10,6
```

In [7]:
```
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
```

In [2]:
```
data=pd.read_csv("AirPassengers.csv")
data['Month']=pd.to_datetime(data['Month'],infer_datetime_format=True)
indexedData=data.set_index(['Month'])
```

In [3]:
```
from datetime import datetime
indexedData.head(5)
```

Out[3]:

Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

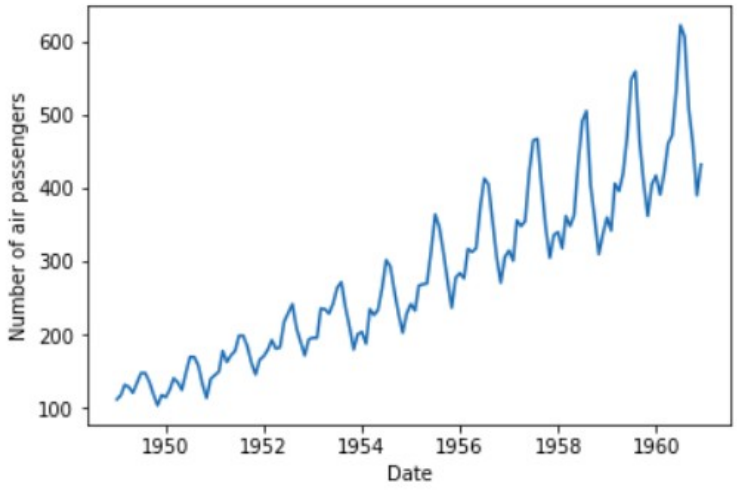
In [4]:
```
indexedData.tail(5)
```

Out[4]:

Month	#Passengers
1960-08-01	606
1960-09-01	508
1960-10-01	461
1960-11-01	390
1960-12-01	432

In [8]:
```
plt.xlabel("Date")
plt.ylabel("Number of air passengers")
plt.plot(indexedData)
```

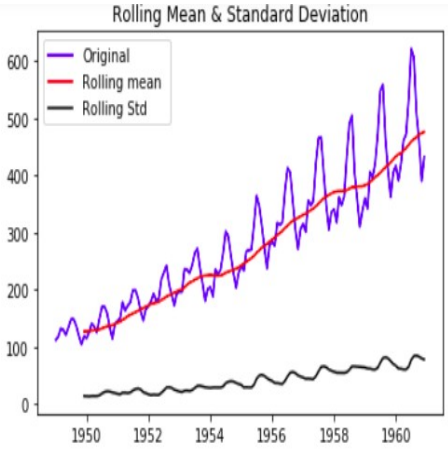
Out[8]:
```
[<matplotlib.lines.Line2D at 0x7fc7e1ec3e48>]
```



In [9]:
```
rolmean=indexedData.rolling(window=12).mean()
rolstd=indexedData.rolling(window=12).std()
print(rolmean,rolstd)
```

```
             #Passengers
Month
1949-01-01           NaN
1949-02-01           NaN
1949-03-01           NaN
1949-04-01           NaN
1949-05-01           NaN
1949-06-01           NaN
1949-07-01           NaN
1949-08-01           NaN
1949-09-01           NaN
1949-10-01           NaN
1949-11-01           NaN
1949-12-01    126.666667
1950-01-01    126.916667
1950-02-01    127.583333
1950-03-01    128.333333
1950-04-01    128.833333
1950-05-01    129.166667
1950-06-01    130.333333
```

In [10]:
```
orig=plt.plot(indexedData,color='blue',label='Original')
mean=plt.plot(rolmean,color='red',label='Rolling mean ')
std=plt.plot(rolstd,color='black',label='Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
```

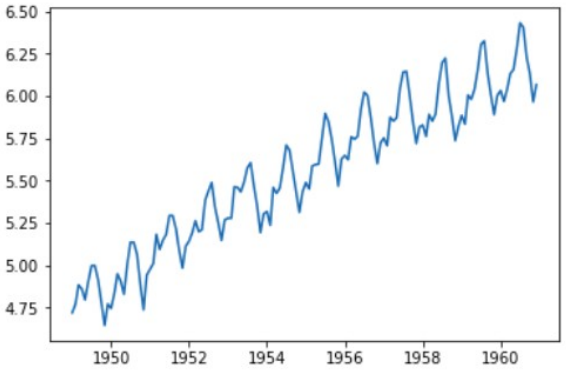


[13]:
```
from statsmodels.tsa.stattools import adfuller

print('Results of Dicky-Fuller Test:')
dftest=adfuller(indexedData['#Passengers'],autolag='AIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print (dfoutput)
```

```
Results of Dicky-Fuller Test:
Test Statistic                  0.815369
p-value                         0.991880
#Lags Used                     13.000000
Number of Observations Used    130.000000
Critical Value (1%)             -3.481682
Critical Value (5%)             -2.884042
Critical Value (10%)            -2.578770
dtype: float64
```

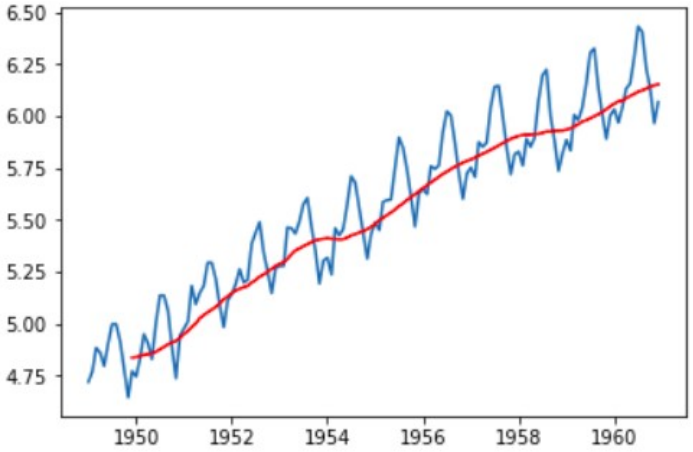
In [14]:
```
#Estimating Trend
idls=np.log(indexedData)
plt.plot(idls)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x7fc7dd0ba710>]
```



In [17]:
```
movingAverage=idls.rolling(window=12).mean()
movingSTD=idls.rolling(window=12).std()
plt.plot(idls)
plt.plot(movingAverage,color='red')
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x7fc7dc350320>]
```



In [18]:
```
datasetLogScaleMinusMovingAverage=idls-movingAverage
datasetLogScaleMinusMovingAverage.head(12)
```

Out[18]:

Month	#Passengers
1949-01-01	NaN
1949-02-01	NaN
1949-03-01	NaN
1949-04-01	NaN
1949-05-01	NaN
1949-06-01	NaN
1949-07-01	NaN
1949-08-01	NaN
1949-09-01	NaN
1949-10-01	NaN
1949-11-01	NaN
1949-12-01	-0.065494

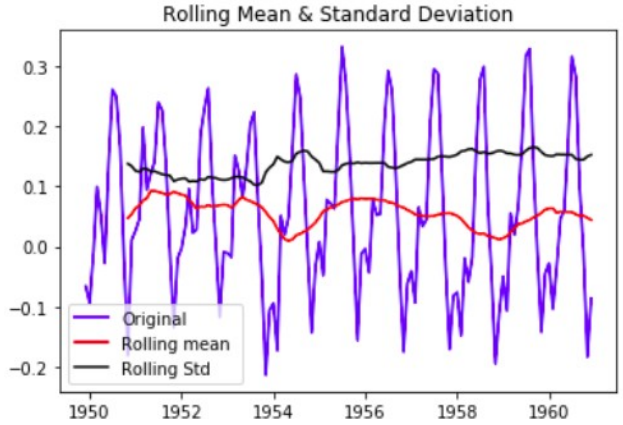
In [19]:
```
#Removing NaN values
datasetLogScaleMinusMovingAverage.dropna(inplace=True)
datasetLogScaleMinusMovingAverage.head(10)
```

Out[19]:

Month	#Passengers
1949-12-01	-0.065494
1950-01-01	-0.093449
1950-02-01	-0.007566
1950-03-01	0.099416
1950-04-01	0.052142
1950-05-01	-0.027529
1950-06-01	0.139881
1950-07-01	0.260184
1950-08-01	0.248635
1950-09-01	0.162937

In [31]:
```
def test_stationarity(timeseries):

    #Rolling statistics
    movingAverage=timeseries.rolling(window=12).mean()
    movingSTD=timeseries.rolling(window=12).std()

    #Plot rolling statistics
    orig=plt.plot(timeseries,color='blue',label='Original')
    mean=plt.plot(movingAverage,color='red',label='Rolling mean ')
    std=plt.plot(movingSTD,color='black',label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dicky-Fuller Test
    print('Results of Dicky-Fuller Test:')
    dftest=adfuller(timeseries['#Passengers'],autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
```

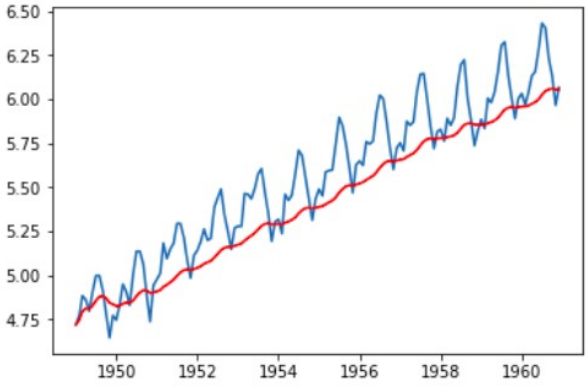
In [32]:
```
test_stationarity(datasetLogScaleMinusMovingAverage)
```



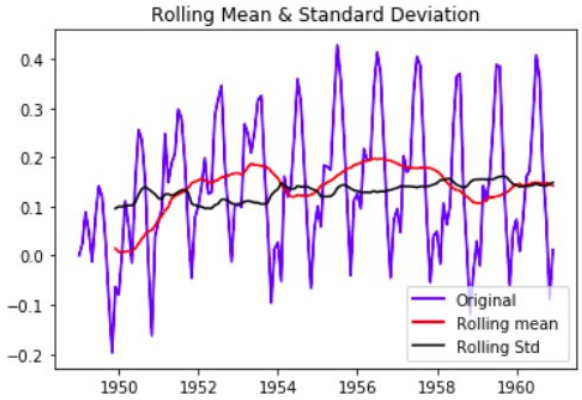

```
Results of Dicky-Fuller Test:
Test Statistic                  -3.162908
p-value                          0.022235
#Lags Used                      13.000000
Number of Observations Used    119.000000
Critical Value (1%)             -3.486535
Critical Value (5%)             -2.886151
Critical Value (10%)            -2.579896
dtype: float64
```

In [33]:
```
expdecayweightedavg=idls.ewm(halflife=12,min_periods=0,adjust=True).mean()
plt.plot(idls)
plt.plot(expdecayweightedavg,color='red')
```

Out[33]:
```
[<matplotlib.lines.Line2D at 0x7fc7d424ea58>]
```



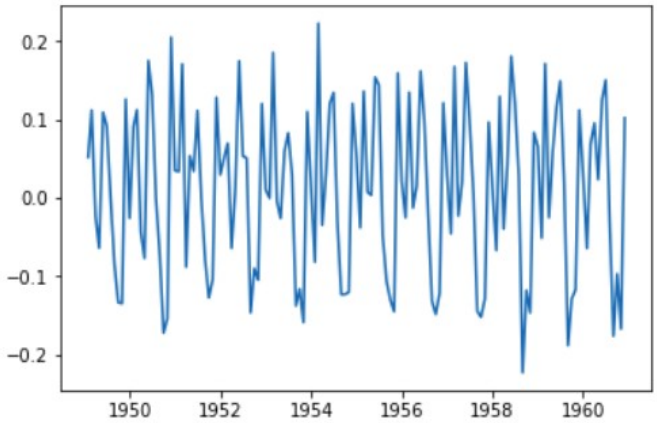
In [34]:
```
datasetLogScaleMinusMovingExponentialDecayAverage=idls-expdecayweightedavg
test_stationarity(datasetLogScaleMinusMovingExponentialDecayAverage)
```



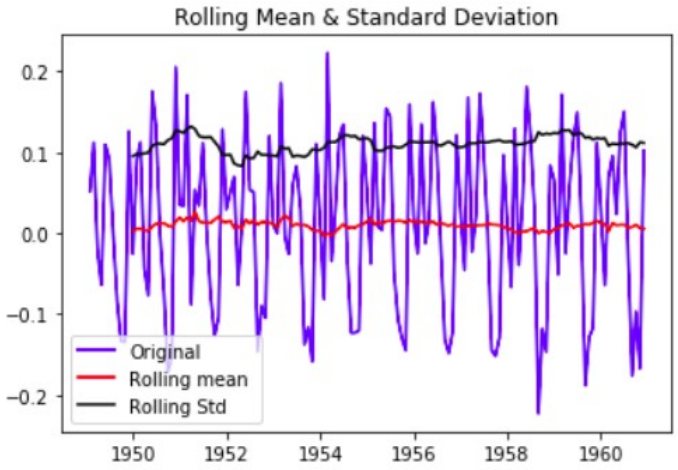

```
Results of Dicky-Fuller Test:
Test Statistic                  -3.601262
p-value                          0.005737
#Lags Used                      13.000000
Number of Observations Used    130.000000
Critical Value (1%)             -3.481682
Critical Value (5%)             -2.884042
Critical Value (10%)            -2.578770
dtype: float64
```

In [35]:
```
datasetLogDiffShifting=idls-idls.shift()
plt.plot(datasetLogDiffShifting)
```

Out[35]:
```
[<matplotlib.lines.Line2D at 0x7fc7d417a940>]
```

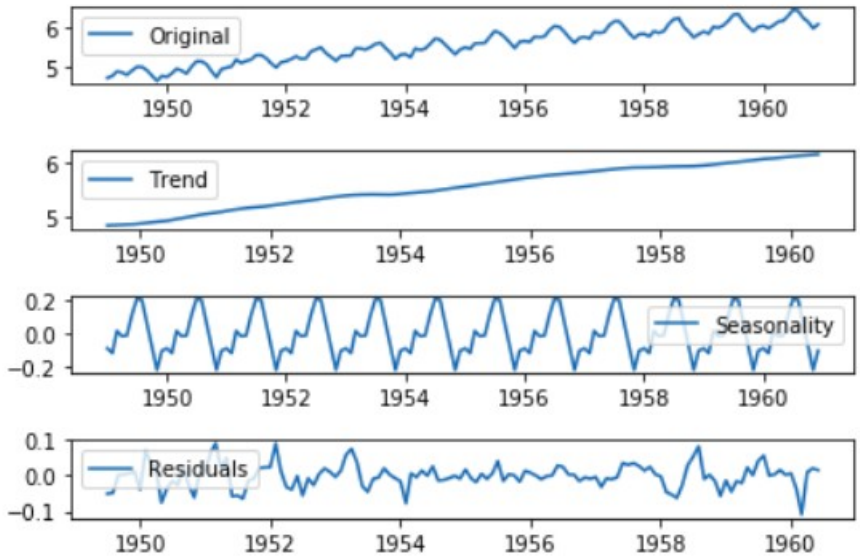


In [36]:
```
datasetLogDiffShifting.dropna(inplace=True)
test_stationarity(datasetLogDiffShifting)
```

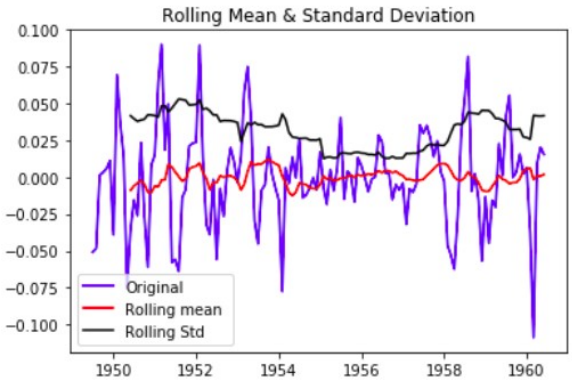



```
Results of Dicky-Fuller Test:
Test Statistic                  -2.717131
p-value                          0.071121
#Lags Used                      14.000000
Number of Observations Used    128.000000
Critical Value (1%)             -3.482501
Critical Value (5%)             -2.884398
Critical Value (10%)            -2.578960
dtype: float64
```

In [38]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(idls)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.subplot(411)
plt.plot(idls, label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
```

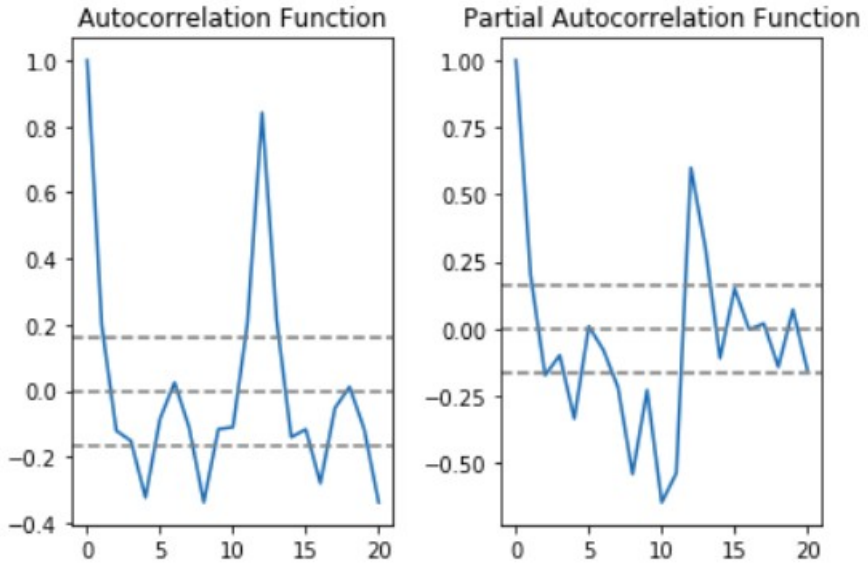


In [39]:
```
decomposedLogdata = residual
decomposedLogdata.dropna(inplace=True)
test_stationarity(decomposedLogdata)
```



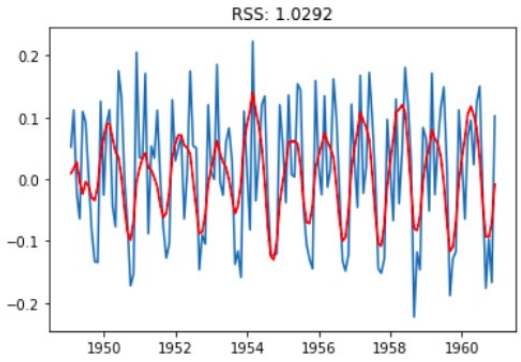

```
Results of Dicky-Fuller Test:
Test Statistic                -6.332387e+00
p-value                        2.885059e-08
#Lags Used                     9.000000e+00
Number of Observations Used    1.220000e+02
Critical Value (1%)           -3.485122e+00
Critical Value (5%)           -2.885538e+00
Critical Value (10%)          -2.579569e+00
dtype: float64
```

In [40]:
```
from statsmodels.tsa.stattools import acf, pacf

lag_acf = acf(datasetLogDiffShifting, nlags=20)
lag_pacf = pacf(datasetLogDiffShifting, nlags=20, method='ols')


#Plot ACF:
plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(datasetLogDiffShifting)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(datasetLogDiffShifting)),linestyle='--',color='gray')
plt.title('Autocorrelation Function')

#Plot PACF:
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(datasetLogDiffShifting)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(datasetLogDiffShifting)),linestyle='--',color='gray')
plt.title('Partial Autocorrelation Function')
plt.tight_layout()
```



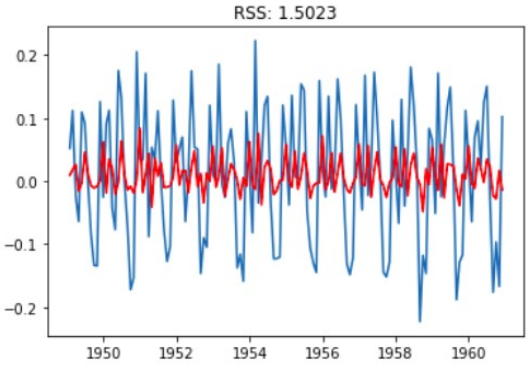
In [47]:
```
from statsmodels.tsa.arima_model import ARIMA

#AR MODEL
model = ARIMA(idls, order=(2, 1, 2))
results_AR = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_AR.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_AR.fittedvalues-datasetLogDiffShifting["#Passengers"])**2))
print('Plotting AR model')
```

```
Plotting AR model
```



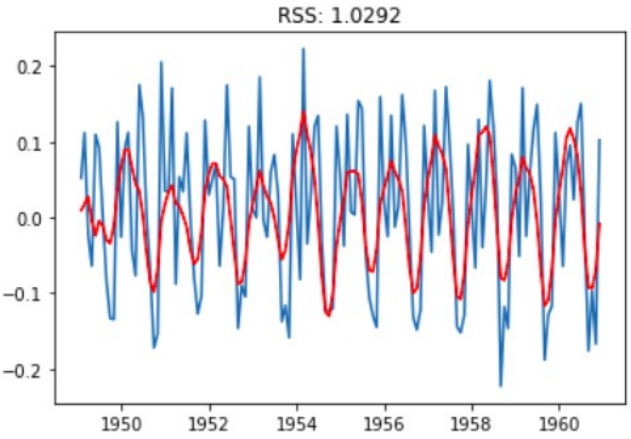
In [44]:
```
#MA MODEL
model = ARIMA(idls, order=(2,1,0))
results_MA = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_MA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_MA.fittedvalues-datasetLogDiffShifting["#Passengers"])**2))
print('Plotting MA model')
```

```
Plotting MA model
```



n [45]:
```
model = ARIMA(idls, order=(2, 1, 2))
results_ARIMA = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_ARIMA.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_ARIMA.fittedvalues-datasetLogDiffShifting["#Passengers"])**2))
```

Out[45]: Text(0.5, 1.0, 'RSS: 1.0292')



In [48]:
```
predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
print (predictions_ARIMA_diff.head())
```

```
Month
1949-02-01    0.009580
1949-03-01    0.017491
1949-04-01    0.027670
1949-05-01   -0.004521
1949-06-01   -0.023890
dtype: float64
```

In [50]:
```
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
print (predictions_ARIMA_diff_cumsum.head())
```

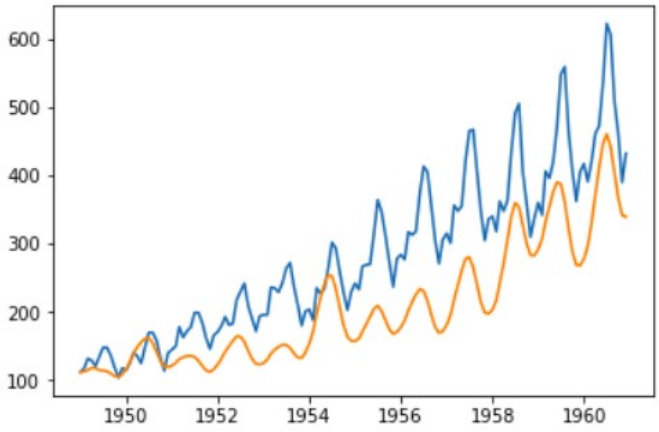
```
Month
1949-02-01    0.009580
1949-03-01    0.027071
1949-04-01    0.054742
1949-05-01    0.050221
1949-06-01    0.026331
dtype: float64
```

In [52]:
```
predictions_ARIMA_log = pd.Series(idls['#Passengers'].iloc[0], index=idls['#Passengers'].index)
predictions_ARIMA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum,fill_value=0)
predictions_ARIMA_log.head()
```

Out[52]:
```
Month
1949-01-01    4.718499
1949-02-01    4.728079
1949-03-01    4.745570
1949-04-01    4.773241
1949-05-01    4.768720
dtype: float64
```

In [54]:
```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(indexedData)
plt.plot(predictions_ARIMA)
```

Out[54]:
```
[<matplotlib.lines.Line2D at 0x7fc7d4633ef0>]
```



In [55]:
```
idls
```

Out[55]:

Month	#Passengers
1949-01-01	4.718499
1949-02-01	4.770685
1949-03-01	4.882802
1949-04-01	4.859812
1949-05-01	4.795791
1949-06-01	4.905275
1949-07-01	4.997212
1949-08-01	4.997212
1949-09-01	4.912655
1949-10-01	4.779123
1949-11-01	4.644391
1949-12-01	4.770685
1950-01-01	4.744932
1950-02-01	4.836282
1950-03-01	4.948760
1950-04-01	4.905275
1950-05-01	4.828314
1950-06-01	5.003946