In [11]:
```
# وارد کردن کتابخانه ها
import numpy
import matplotlib.pyplot as plt
import pandas
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

In [12]:
```
# load the dataset
dataset = pandas.read_csv( 'https://raw.githubusercontent.com/jbrownlee/Data
dataset = dataset.values.astype('float32')
```

In [13]:
```
dataset
```

Out[13]:
```
array([[112.],
       [118.],
       [132.],
       [129.],
       [121.],
       [135.],
       [148.],
       [148.],
       [136.],
       [119.],
       [104.],
       [118.],
       [115.],
       [126.],
       [141.],
       [135.],
       [125.],
       [149.],
       [170.],
```

In [14]:
```
# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
```

In [15]:
```
1 dataset
```

Out[15]:
```
array([[0.01544401],
       [0.02702703],
       [0.05405405],
       [0.04826255],
       [0.03281853],
       [0.05984557],
       [0.08494207],
       [0.08494207],
       [0.06177607],
       [0.02895753],
       [0.        ],
       [0.02702703],
       [0.02123553],
       [0.04247104],
       [0.07142857],
       [0.05984557],
       [0.04054055],
       [0.08687258],
       [0.12741312],
```

In [16]:
```
1 # split into train and test sets
2 train_size = int(len(dataset) * 0.67)
3 test_size = len(dataset) - train_size
4 train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
```

In [17]:
```
1 def create_dataset(dataset, look_back=1):
2     dataX, dataY = [], []
3     for i in range(len(dataset)-look_back-1):
4         a = dataset[i:(i+look_back), 0]
5         dataX.append(a)
6         dataY.append(dataset[i + look_back, 0])
7     return numpy.array(dataX), numpy.array(dataY)
```

In [18]:
```
1 # reshape into X=t and Y=t+1
2 look_back = 1
3 trainX, trainY = create_dataset(train, look_back)
4 testX, testY = create_dataset(test, look_back)
```

In [19]:
```
1 # reshape input to be [samples, time steps, features]
2 trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
3 testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

In [10]:
```
1 trainX
```

Out[10]:
```
array([[[0.01544401]],

       [[0.02702703]],

       [[0.05405405]],

       [[0.04826255]],

       [[0.03281853]],

       [[0.05984557]],

       [[0.08494207]],

       [[0.08494207]],

       [[0.06177607]],

       [[0.02895753]],
```

In [20]:
```
1 # create and fit the LSTM network
2 model = Sequential()
3 model.add(LSTM(4, input_dim=look_back))
4 model.add(Dense(1))
```

```
C:\Users\ShahinN\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarni
ng: The `input_dim` and `input_length` arguments in recurrent layers are deprec
ated. Use `input_shape` instead.
  This is separate from the ipykernel package so we can avoid doing imports unt
il
C:\Users\ShahinN\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarni
ng: Update your `LSTM` call to the Keras 2 API: `LSTM(4, input_shape=(None, 1))
`
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```

In [21]:
```
model.compile(loss= 'mean_squared_error' , optimizer= 'adam' )
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)
```

```
WARNING:tensorflow:From C:\Users\ShahinN\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/100
 - 2s - loss: 0.0502
Epoch 2/100
 - 0s - loss: 0.0267
Epoch 3/100
 - 0s - loss: 0.0201
Epoch 4/100
 - 0s - loss: 0.0181
Epoch 5/100
 - 0s - loss: 0.0172
Epoch 6/100
 - 0s - loss: 0.0164
Epoch 7/100
 - 0s - loss: 0.0157
Epoch 8/100
```

In [17]:
```
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
```

In [18]:
```
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
```

In [19]:
```
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print( 'Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print( 'Test Score: %.2f RMSE' % (testScore))
```

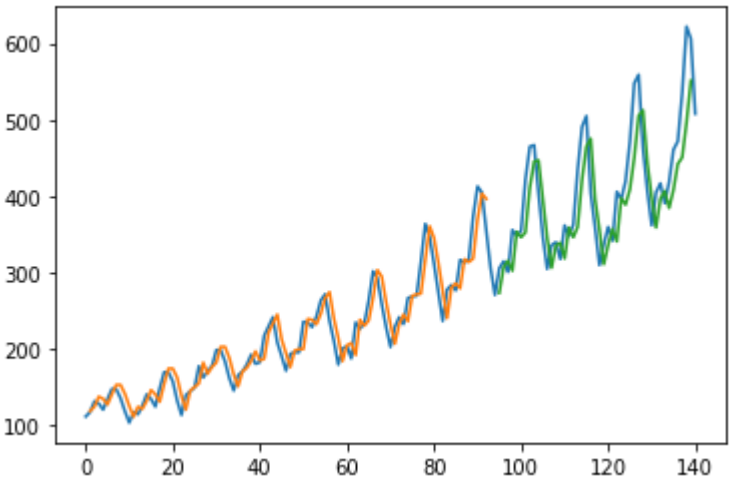
```
Train Score: 22.17 RMSE
Test Score: 48.07 RMSE
```

In [20]:
```
# shift train predictions for plotting
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict


# shift test predictions for plotting
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :] = testP



# plot baseline and predictions
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```




In [ ]:
```
1
```