In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
Data=pd.read_excel(r"/content/Crude Oil Prices Daily.xlsx")
Data
```

Out[2]:

	Date	Closing Value
0	**1986-01-02**	**25.56**
1	**1986-01-03**	**26.00**
2	**1986-01-06**	**26.53**
3	**1986-01-07**	**25.85**
4	**1986-01-08**	**25.87**
...	**...**	**...**
8218	**2018-07-03**	**74.19**
8219	**2018-07-04**	**NaN**
8220	**2018-07-05**	**73.05**
8221	**2018-07-06**	**73.78**
8222	**2018-07-09**	**73.93**

8223 rows × 2 columns

In [3]:

```
Data.isnull().any()
```

Out[3]:

```
Date             False
Closing Value     True
dtype: bool
```

In [4]:

```
Data.isnull().sum()
```

Out[4]:

```
Date             0
Closing Value    7
dtype: int64
```

In [5]:

```
Data.dropna(axis=0,inplace=True)
Data.isnull().sum()
```

Out[5]:

```
Date             0
Closing Value    0
dtype: int64
```

In [6]:

```
Data_oil=Data.reset_index()['Closing Value']
```

In [7]:

```
Data_oil
```

Out[7]:

```
0       25.56
1       26.00
2       26.53
3       25.85
4       25.87
        ...
8211    73.89
8212    74.19
8213    73.05
8214    73.78
8215    73.93
Name: Closing Value, Length: 8216, dtype: float64
```

In [8]:

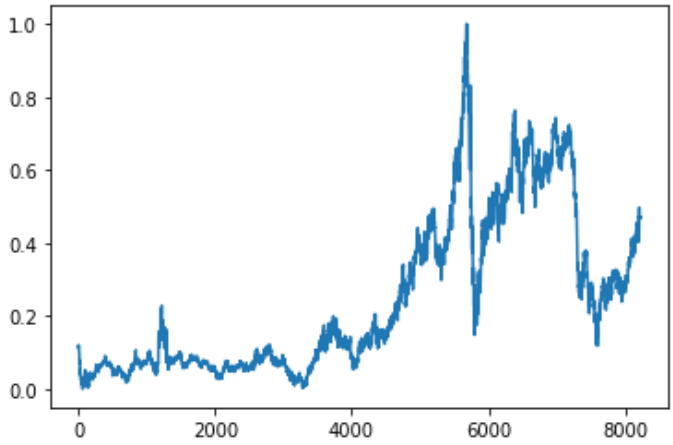
```
from sklearn.preprocessing import MinMaxScaler
scaler= MinMaxScaler(feature_range=(0,1))
Data_oil=scaler.fit_transform(np.array(Data_oil).reshape(-1,1))
```

In [9]:

```
plt.plot(Data_oil)
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7f27b218f9d0>]
```



In [10]:

```
training_size=int(len(Data_oil)*0.65)
test_size=len(Data_oil)-training_size
train_data,test_Data=Data_oil[0:training_size,:],Data_oil[training_size:len(Data_oil),:1]
```

In [11]:

```
training_size,test_size
```

Out[11]:

```
(5340, 2876)
```

In [12]:

```
train_data.shape
```

Out[12]:

```
(5340, 1)
```

In [13]:

```
def create_dataset(dataset, time_step=1):
  dataX, dataY = [],[]
  for i in range(len(dataset)-time_step-1):
    a = dataset[i:(i+time_step),0]
    dataX.append(a)
    dataY.append(dataset[i + time_step, 0])
  return np.array(dataX),np.array(dataY)
```

In [14]:

```
time_step = 10
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_Data, time_step)
```

In [15]:

```
print(X_train.shape),print(y_train.shape)
```

```
(5329, 10)
(5329,)
```

Out[15]:

```
(None, None)
```

In [16]:

```
print(X_test.shape),print(ytest.shape)
```

```
(2865, 10)
(2865,)
```

Out[16]:

```
(None, None)
```

In [17]:

```
X_train
```

Out[17]:

```
array([[0.11335703, 0.11661484, 0.12053902, ..., 0.10980305, 0.1089886 ,
        0.11054346],
       [0.11661484, 0.12053902, 0.11550422, ..., 0.1089886 , 0.11054346,
        0.10165852],
       [0.12053902, 0.11550422, 0.1156523 , ..., 0.11054346, 0.10165852,
        0.09906708],
       ...,
       [0.36731823, 0.35176958, 0.36080261, ..., 0.36391234, 0.37042796,
        0.37042796],
       [0.35176958, 0.36080261, 0.35354657, ..., 0.37042796, 0.37042796,
        0.37879461],
       [0.36080261, 0.35354657, 0.35295424, ..., 0.37042796, 0.37879461,
        0.37916482]])
```

In [18]:

```
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1],1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1], 1)
```

Model building

In [19]:

```
import tensorflow
import keras
```

In [20]:

```
from keras.models import Sequential
```

```
from keras.layers import Dense
from keras.layers import LSTM
```

In [21]:

```
model=Sequential()
```

In [22]:

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(10,1)))
model.add(LSTM(50, return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
```

In [23]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 50)            10400

 lstm_1 (LSTM)               (None, 10, 50)            20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

```
model.compile(loss='mean_squared_error',optimizer='adam')
```

In [45]:

```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=50,batch_size=64,verbose
=1)
```

```
Epoch 1/50
84/84 [==============================] - 2s 26ms/step - loss: 3.3717e-05 - val_loss: 2.21
85e-04
Epoch 2/50
84/84 [==============================] - 2s 25ms/step - loss: 3.8558e-05 - val_loss: 5.46
94e-04
Epoch 3/50
84/84 [==============================] - 2s 25ms/step - loss: 3.1944e-05 - val_loss: 1.86
47e-04
Epoch 4/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5854e-05 - val_loss: 2.58
25e-04
Epoch 5/50
84/84 [==============================] - 2s 25ms/step - loss: 3.6433e-05 - val_loss: 2.80
71e-04
Epoch 6/50
84/84 [==============================] - 2s 25ms/step - loss: 3.3365e-05 - val_loss: 2.76
71e-04
Epoch 7/50
84/84 [==============================] - 2s 25ms/step - loss: 3.3240e-05 - val_loss: 2.02
31e-04
Epoch 8/50
84/84 [==============================] - 2s 25ms/step - loss: 3.2062e-05 - val_loss: 4.78
69e-04
Epoch 9/50
```

84/84 [==============================] - 2s 25ms/step - loss: 3.4865e-05 - val_loss: 3.95
79e-04
Epoch 10/50
84/84 [==============================] - 2s 25ms/step - loss: 3.3310e-05 - val_loss: 2.04
87e-04
Epoch 11/50
84/84 [==============================] - 2s 25ms/step - loss: 3.4019e-05 - val_loss: 1.87
11e-04
Epoch 12/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5673e-05 - val_loss: 1.82
75e-04
Epoch 13/50
84/84 [==============================] - 2s 26ms/step - loss: 3.1530e-05 - val_loss: 2.65
82e-04
Epoch 14/50
84/84 [==============================] - 2s 25ms/step - loss: 3.1700e-05 - val_loss: 2.02
30e-04
Epoch 15/50
84/84 [==============================] - 2s 26ms/step - loss: 3.3366e-05 - val_loss: 1.94
67e-04
Epoch 16/50
84/84 [==============================] - 2s 26ms/step - loss: 3.2785e-05 - val_loss: 1.95
43e-04
Epoch 17/50
84/84 [==============================] - 2s 25ms/step - loss: 3.7705e-05 - val_loss: 1.79
21e-04
Epoch 18/50
84/84 [==============================] - 2s 25ms/step - loss: 3.4293e-05 - val_loss: 1.95
76e-04
Epoch 19/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5124e-05 - val_loss: 2.09
53e-04
Epoch 20/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5138e-05 - val_loss: 1.79
12e-04
Epoch 21/50
84/84 [==============================] - 2s 26ms/step - loss: 3.0977e-05 - val_loss: 2.41
47e-04
Epoch 22/50
84/84 [==============================] - 2s 25ms/step - loss: 3.2895e-05 - val_loss: 2.80
17e-04
Epoch 23/50
84/84 [==============================] - 2s 25ms/step - loss: 3.1144e-05 - val_loss: 1.90
05e-04
Epoch 24/50
84/84 [==============================] - 2s 24ms/step - loss: 3.0762e-05 - val_loss: 4.52
58e-04
Epoch 25/50
84/84 [==============================] - 2s 25ms/step - loss: 3.4665e-05 - val_loss: 1.72
32e-04
Epoch 26/50
84/84 [==============================] - 2s 25ms/step - loss: 2.9580e-05 - val_loss: 3.01
59e-04
Epoch 27/50
84/84 [==============================] - 2s 25ms/step - loss: 3.3668e-05 - val_loss: 1.82
48e-04
Epoch 28/50
84/84 [==============================] - 2s 25ms/step - loss: 3.6942e-05 - val_loss: 1.91
43e-04
Epoch 29/50
84/84 [==============================] - 2s 25ms/step - loss: 3.3591e-05 - val_loss: 1.82
73e-04
Epoch 30/50
84/84 [==============================] - 2s 25ms/step - loss: 3.6756e-05 - val_loss: 2.02
32e-04
Epoch 31/50
84/84 [==============================] - 2s 25ms/step - loss: 3.6306e-05 - val_loss: 1.80
30e-04
Epoch 32/50
84/84 [==============================] - 2s 25ms/step - loss: 3.2851e-05 - val_loss: 2.48
47e-04
Epoch 33/50
84/84 [==============================] - 2s 24ms/step - loss: 3.0010e-05 - val_loss: 1.81

```
84/84 [==============================] - 2s 24ms/step - loss: 3.0910e-05 - val_loss: 1.81
17e-04
Epoch 34/50
84/84 [==============================] - 2s 25ms/step - loss: 3.4033e-05 - val_loss: 2.13
45e-04
Epoch 35/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5577e-05 - val_loss: 1.68
30e-04
Epoch 36/50
84/84 [==============================] - 2s 25ms/step - loss: 2.9806e-05 - val_loss: 2.56
35e-04
Epoch 37/50
84/84 [==============================] - 2s 25ms/step - loss: 3.0922e-05 - val_loss: 1.87
32e-04
Epoch 38/50
84/84 [==============================] - 2s 26ms/step - loss: 2.9814e-05 - val_loss: 1.87
01e-04
Epoch 39/50
84/84 [==============================] - 2s 25ms/step - loss: 3.0128e-05 - val_loss: 2.94
89e-04
Epoch 40/50
84/84 [==============================] - 2s 25ms/step - loss: 3.2662e-05 - val_loss: 1.83
38e-04
Epoch 41/50
84/84 [==============================] - 2s 26ms/step - loss: 3.1931e-05 - val_loss: 2.57
32e-04
Epoch 42/50
84/84 [==============================] - 2s 25ms/step - loss: 3.2837e-05 - val_loss: 1.84
84e-04
Epoch 43/50
84/84 [==============================] - 2s 25ms/step - loss: 3.3000e-05 - val_loss: 2.09
29e-04
Epoch 44/50
84/84 [==============================] - 2s 25ms/step - loss: 3.2694e-05 - val_loss: 1.76
41e-04
Epoch 45/50
84/84 [==============================] - 2s 25ms/step - loss: 3.0530e-05 - val_loss: 1.87
18e-04
Epoch 46/50
84/84 [==============================] - 2s 25ms/step - loss: 3.0992e-05 - val_loss: 1.65
74e-04
Epoch 47/50
84/84 [==============================] - 2s 25ms/step - loss: 2.9559e-05 - val_loss: 2.41
66e-04
Epoch 48/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5584e-05 - val_loss: 1.84
56e-04
Epoch 49/50
84/84 [==============================] - 2s 25ms/step - loss: 3.0129e-05 - val_loss: 2.05
35e-04
Epoch 50/50
84/84 [==============================] - 2s 25ms/step - loss: 3.0142e-05 - val_loss: 2.02
75e-04
```

Out[45]:

```
<keras.callbacks.History at 0x7f274f03afd0>
```

In [46]:

```
from sklearn.model_selection import train_test_split
```

In [47]:

```
import tensorflow as tf
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)
```

```
167/167 [==============================] - 1s 6ms/step
90/90 [==============================] - 1s 6ms/step
```

In [48]:

```
train_predict= scaler.inverse_transform(train_predict)
test_predict=  scaler.inverse_transform(test_predict)
```

In [49]:

```
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

Out[49]:

```
29.578497989297606
```

In [31]:

```
math.sqrt(mean_squared_error(ytest,test_predict))
```

Out[31]:

```
78.11974774364951
```

In [32]:

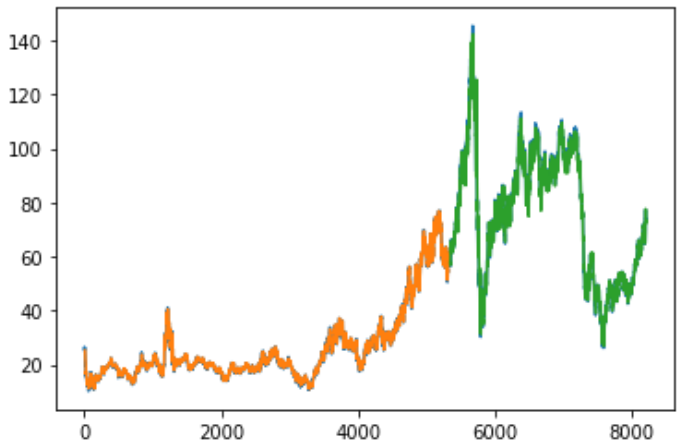
```
from tensorflow.keras.models import load_model
```

In [33]:

```
model.save("crude_oil.h5")
```

In [34]:

```
look_back=10
trainPredictPlot = np.empty_like(Data_oil)
trainPredictPlot[:, :]=np.nan
trainPredictPlot[look_back:len(train_predict)+look_back,:]= train_predict
testPredictPlot = np.empty_like(Data_oil)
testPredictPlot[:, :]=np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(Data_oil)-1, :]= test_predict
plt.plot(scaler.inverse_transform(Data_oil))
plt.plot(trainPredictPlot,label="traindata")
plt.plot(testPredictPlot,label="testdata")
plt.show()
print("Green indicates predicated data")
print("Blue indicates complete data")
print("Orange indicates train data")
```




```
Green indicates predicated data
Blue indicates complete data
Orange indicates train data
```

In [50]:

```
len(test_Data)
```

Out[50]:

```
2876
```

In [51]:

```
X_input=test_Data[2866:].reshape(1,-1)
X_input.shape
```

Out[51]:

```
(1, 10)
```

In [37]:

```
temp_input=list(X_input)
temp_input=temp_input[0].tolist()
```

In [52]:

```
temp_input
```

Out[52]:

```
[0.47149415074781587,
 0.4708293676376343,
 0.47109171748161316,
 0.4705319106578827,
 0.4691915214061737,
 0.467776894569397,
 0.46619293093681335,
 0.46466758847236633,
 0.46324965357780457,
 0.46217411756515503,
 0.461301416158676151]
```

In [39]:

```
lst_output=[]
n_steps=10
i=0
while(i<10):
  if(len(temp_input)>10):
    X_input=np.array(temp_input[1:])
    print("{} Day input {}".format(i,X_input))
    X_input=X_input.reshape(1,-1)
    X_input=X_input.reshape((1,n_steps,1))
    yhat=model.predict(X_input, verbose=0)
    print("{} Day output {}".format(i,yhat))
    temp_input.extend(yhat[0].tolist())
    temp_input=temp_input[1:]
    lst_output.extend(yhat.tolist())
    i=i+1
  else:
    X_input=X_input.reshape((1,n_steps,1))
    yhat=model.predict(X_input, verbose=0)
    print(yhat[0])
    temp_input.extend(yhat[0].tolist())
    print(len(temp_input))
    lst_output.extend(yhat.tolist())
    i=i+1
```

```
[0.47082937]
11
1 Day input [0.4811195  0.49726048 0.46794017 0.47297497 0.47119799 0.47341922
 0.46497853 0.47038353 0.47149415 0.47082937]
1 Day output [[0.47109172]]
2 Day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922 0.46497853
 0.47038353 0.47149415 0.47082937 0.47109172]
2 Day output [[0.4705319]]
3 Day input [0.46794017 0.47297497 0.47119799 0.47341922 0.46497853 0.47038353
 0.47149415 0.47082937 0.47109172 0.47053191]
3 Day output [[0.46919152]]
4 Day input [0.47297497 0.47119799 0.47341922 0.46497853 0.47038353 0.47149415
 0.47082937 0.47109172 0.47053191 0.46919152]
```

```
4 Day output [[0.4677769]]
5 Day input [0.47119799 0.47341922 0.46497853 0.47038353 0.47149415 0.47082937
 0.47109172 0.47053191 0.46919152 0.46777689]
5 Day output [[0.46619293]]
6 Day input [0.47341922 0.46497853 0.47038353 0.47149415 0.47082937 0.47109172
 0.47053191 0.46919152 0.46777689 0.46619293]
6 Day output [[0.4646676]]
7 Day input [0.46497853 0.47038353 0.47149415 0.47082937 0.47109172 0.47053191
 0.46919152 0.46777689 0.46619293 0.46466759]
7 Day output [[0.46324965]]
8 Day input [0.47038353 0.47149415 0.47082937 0.47109172 0.47053191 0.46919152
 0.46777689 0.46619293 0.46466759 0.46324965]
8 Day output [[0.46217412]]
9 Day input [0.47149415 0.47082937 0.47109172 0.47053191 0.46919152 0.46777689
 0.46619293 0.46466759 0.46324965 0.46217412]
9 Day output [[0.46130142]]
```

In [40]:

```
day_new=np.arange(1,11)
day_pred=np.arange(11,21)
```

In [41]:

```
len(Data_oil)
```

Out[41]:

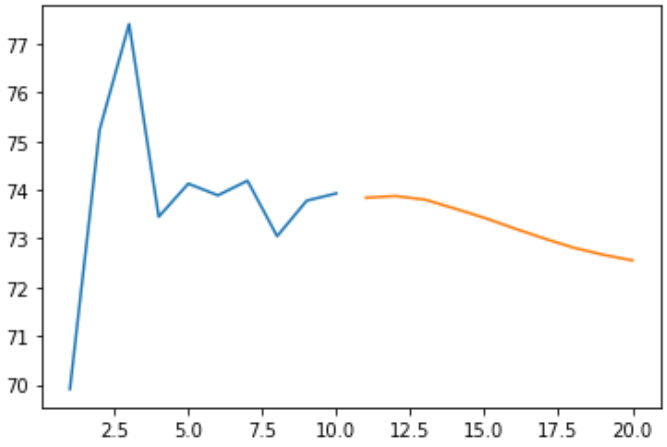
```
8216
```

In [42]:

```
plt.plot(day_new,scaler.inverse_transform(Data_oil[8206:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x7f274f095ad0>]
```

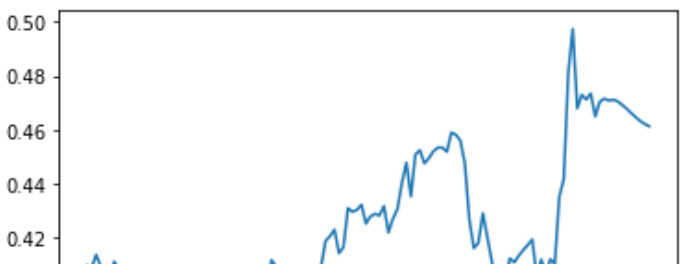


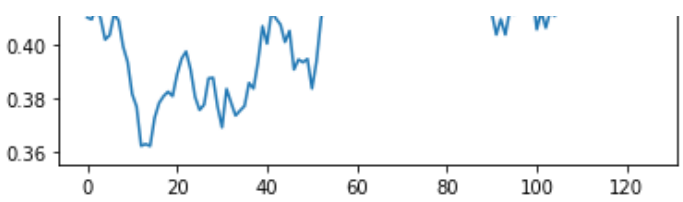
In [43]:

```
df3=Data_oil.tolist()
df3.extend(lst_output)
plt.plot(df3[8100:])
```

Out[43]:

```
[<matplotlib.lines.Line2D at 0x7f274f016d90>]
```





In [44]:

```
df3=scaler.inverse_transform(df3).tolist()
plt.plot(df3)
```

Out[44]:

```
[<matplotlib.lines.Line2D at 0x7f274bb781d0>]
```