Program 6:

Design and implement a deep learning network for forecasting time series data.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Load and scale data
df = pd.read_csv('data.csv', index_col=0, parse_dates=True)
data = df['value'].values.reshape(-1, 1)
scaler = MinMaxScaler((0, 1))
scaled = scaler.fit_transform(data)

# Create dataset
def create_dataset(data, window):
    X, Y = [], []
    for i in range(len(data) - window):
        X.append(data[i:i+window, 0])
        Y.append(data[i+window, 0])
    return np.array(X), np.array(Y)

window = 60
X, Y = create_dataset(scaled, window)
X = X.reshape(-1, window, 1)

# Split
```

```
split = int(len(X) * 0.8)
X_train, X_test, Y_train, Y_test = X[:split], X[split:], Y[:split], Y[split:]

# Model
model = Sequential([
    LSTM(50, activation='relu', input_shape=(window, 1)),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, Y_train, epochs=30, batch_size=32, verbose=1)

# Evaluate
pred = scaler.inverse_transform(model.predict(X_test))
Y_test_inv = scaler.inverse_transform(Y_test.reshape(-1, 1))
rmse = np.sqrt(mean_squared_error(Y_test_inv, pred))
print(f"Test RMSE: {rmse:.2f}")

# Forecast
future_steps = 15
last_seq = scaled[-window:].reshape(1, window, 1)
forecast = []

for _ in range(future_steps):
    next_val = model.predict(last_seq)[0, 0]
    forecast.append(next_val)
    last_seq = np.append(last_seq[:, 1:, :], [[next_val]], axis=1)

forecast = scaler.inverse_transform(np.array(forecast).reshape(-1, 1))
```

```
print("Future forecast:", forecast.flatten())
```