### **Assignment: Comparing Weather Forecasts for Two Cities**

#### **Objective:**

Develop a Spring application that compares the weather forecast for the next 7 days between two cities using the Open-Meteo API and Spring REST client. Create an Excel spreadsheet to map the City Names and the Location Coordinates. Using the location coordinates fetch the weather forecast Using Open-Meteo Opensource Api. (No api keys are required )

#### **Requirements:**

1. **Project Setup:**
   1. Create a new Spring project using Spring Initializr.
   2. Include dependencies for Spring Rest Template.
2. **Entity Class:**
   1. Create a simple POJO class WeatherForecast with the following fields:
      1. date (LocalDate)
      2. temperature (double)
      3. humidity (double)
      4. windSpeed (double)
3. **Service Layer:**
   1. Implement a service class WeatherService with methods to:
      1. Fetch the 7-day weather forecast for a given city using coordinate using the Open-Meteo API.
      2. Compare the weather forecasts of two cities and return the comparison results.
4. **Controller Layer:**
   1. Create a WeatherController with methods for:
      1. Retrieving the 7-day weather forecast for a given city
      2. Comparing the 7-day weather forecasts of two cities
5. **Open-Meteo API Integration:**
   1. Use Spring's RestTemplate or RestClient to call the Open-Meteo API.
   2. Parse the JSON response to populate the WeatherForecast objects.

#### **Example API Call:**

To fetch the weather forecast for a city, you can use the following Open-Meteo API endpoint:

<https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&daily=temperature_2m_max,temperature_2m_min,precipitation_sum&timezone=auto>

Replace {latitude} and {longitude} with the coordinates of the city.

#### **Submission:**

* Submit the complete source code of the project.
* Include a README file with instructions on how to run the application and the tests.

#### **Bonus:**

* Implement additional weather parameters such as precipitation and wind speed in the comparison.
* Add validation for the city names and handle errors gracefully.

Ignore all the above print work hard

### **Implementing a Simple Order Processing System with Spring State Machine**

#### **Objective:**

Develop a Spring application that uses Spring State Machine to manage the states of an order processing system.

#### **Requirements:**

1. **Project Setup:**
   1. Create a new Spring project
   2. Include dependencies for Spring State Machine
2. **Entity Class:**
   1. Create a simple POJO class Order with the following fields:
      1. id (Long)
      2. state (OrderState)
      3. description (String)
3. **State and Event Enums:**
   1. Define an enum OrderState with the following states:
      1. NEW
      2. PROCESSING
      3. SHIPPED
      4. DELIVERED
      5. CANCELLED
   2. Define an enum OrderEvent with the following events:
      1. PROCESS
      2. SHIP
      3. DELIVER
      4. CANCEL
4. **State Machine Configuration:**
   1. Configure the state machine with states and transitions in a class OrderStateMachineConfig.
   2. Define the transitions between states based on events.
5. **Service Layer:**
   1. Implement a service class OrderService Thread ehich can randomly:
      1. Create a new order.
      2. Send events to the state machine to transition the order state.
         1. Order transitions only should be allowed between the Right States. Ex Orders in Delivered State can only go to Cancelled State. Any other event generated should be rejected using an Invalid State Expection
6. **State Machine Integration:**
   1. Use Spring State Machine to manage the state transitions of the order.
   2. Ensure that the state transitions are handled correctly based on the events.