**Overview**

A weather generator produces a “synthetic” time series of weather data for a location based on  
the statistical characteristics of observed weather at that location. You can think of a weather  
generator as being a simulator of future weather based on observed past weather. A time series is  
a collection of observations generated sequentially through time. The special feature of a time  
series is that successive observations are usually expected to be dependent. In fact, this dependence  
is often exploited in forecasting.

Since we are just beginning as weather forecasters, we will simplify our predictions to just  
whether measurable precipitation will fall from the sky. If there is measurable precipitation,  
we call it a “wet” day. Otherwise, we call it a “dry” day.

**Weather Persistence**

To help with understanding relationships and sequencing events through time, here’s a simple  
pseudocode that shows what it means for precipitation to be persistent from one day to the next.

READ "Did it rain today?"

IF answer is yes THEN

There is a greater chance that tomorrow will be wet rather than dry

ELSE

There is a greater chance that tommorrow will be dry rather than wet

ENDIF

YouTube has literally thousands of videos about weather fronts and how they are connected to weather.

[This one](https://youtu.be/G7Ewqm0YHUI) from the UK has graphics that are supportive of the idea of persistence (though that word is not used).

As you watch it, consider that whatever is causing weather today (high pressure and a warm mass of air

creating a sunny, warm day or low pressure and a cool mass of air creating a cloudy, cool day) is possibly

still going to be affecting weather tomorrow.  This is the idea of persistence.

**Time of year and location**

Weather data depends on both the time of year and the location. This means that the probabilities

used in the simulation need to be associated with both a location and a time of year.

The table below lists the probabilities that a day will be wet given that the previous day was dry

for each month for a weather station near Norman, OK. This table gives the probability of a change

from dry to wet. These are “real” numbers that reflect how often the weather changed from dry to wet

in that specific location, during the month indicated, over the 30-year period from 1970-2000.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| January | February | March | April | May | June | July | August | September | October | November | December |
| 0.27 | 0.33 | 0.40 | 0.46 | 0.43 | 0.28 | 0.12 | 0.17 | 0.23 | 0.21 | 0.28 | 0.27 |

The next table lists the probabilities that a day will be wet given that the previous day was wet for

the same weather station near Norman, OK. This table gives the probability that the weather remains

wet from one day to the next.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| January | February | March | April | May | June | July | August | September | October | November | December |
| 0.55 | 0.58 | 0.61 | 0.69 | 0.73 | 0.62 | 0.45 | 0.55 | 0.58 | 0.55 | 0.59 | 0.55 |

Armed with these probabilities, we can turn our simulation into a weather generator for this location.

Here’s what it would look like for July in Norman, OK.

READ "Did it rain today?"

IF answer is yes THEN

READ a random value between 0 and 1

IF the random value is less than or equal to 0.45 THEN

No change! It is a wet day

ELSE

Change! It is a dry day

ENDIF

ELSE

READ a random value between 0 and 1

IF the random value is less than or equal 0.12 THEN

Change! It is a wet day

ELSE

No change! It is a dry day

ENDIF

ENDIF

If it’s a dry day, we want the outcome to simulate “no change” 88% of the time and “change” 12% of the time.

A common practice would be to use a random number generator to generate some value between 0 and 1. If the

random value is less than .88, then there would be no change, and if it is greater than .88 then the weather

changes to rain.

If it’s a wet day, we want to simulate “no change” 45% of the time and “change” 55% of the time. To do this

with our random number generator, we say there is “no change” if random number is less than .45 and a change

to dry if it is greater.

**Weather generator**

Now it’s time to generate some weather!

Imagine you are a farmer. Does knowing the number of wet or dry days tell the whole story? Would the pattern

be important? If so, what pattern would you like to see? How would you measure this pattern?

The transition probabilities that we have used for Norman, OK are based on historical data, and you might

use them to get a sense for the likelihood certain weather phenomena in the near future. For instance, a

farmer might want to run many, many simulations to get an idea of the likelihood of going 20 or more days

without rain, and the results might influence the crops that he or she plants.

Just as we can base the transition probabilities on historical data, we can also base them on future predictions.

For instance, the National Center for Atmospheric Research (NCAR) simulates weather as it responds to assumptions

about how various “forcings” (e.g, greenhouse gasses) will evolve in the future. Typically, these models couple

an atmospheric model with an ocean model, but more recent versions, the so-called Earth system models, incorporate

more components including land use, sea and land ice, etc. The models can be used to predict future precipitation

patterns and transition probabilities that are based on these forecasts, rather than past data.

The weather generator methods you will be writing for this assignment will:

1. predict future precipitation pattern for one month: oneMonthGenerator
2. find the number of wet or dry days in a given month’s forecast: numberOfWetDryDays
3. find the longest wet or dry spell in a given month’s forecast: lengthOfLongestWetDrySpell

**Future transition probability table as a 2D array**

The oneMonthGenerator method receives as arguments the transition probability tables (dry to wet, and wet to wet) as 2D arrays.

Each table row corresponds to a location (longitude, latitude) in the USA and contains the transition probabilities

for each month of the year.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Longitude | Latitude | January | February | March | April | May | June | July | August | September | October | November | December |
| -97.58 | 26.02 | 0.76 | 0.75 | 0.77 | 0.74 | 0.80 | 0.86 | 0.94 | 0.97 | 0.89 | 0.77 | 0.74 | 0.77 |

Following are the methods to be completed in WeatherGenerator.java:

**public class WeatherGenerator {**

/\* Given a location (longitude, latitude) in the USA and a month of the year, the method

\* returns the forecast for the month based on the drywet and wetwet transition

\* probabilities tables.

\*

\* month will be a value between 2 and 13: 2 corresponds to January, 3 corresponds to February

\* and so on. These are the column indexes of each month in the transition probabilities tables.

\*

\* The first day of the month has a 50% chance to be a wet day, 0-0.49 (wet), 0.50-0.99 (dry)

\*  
 \* Use StdRandom.uniform() to generate a real number uniformly in [0,1)  
 \*/

**int[] oneMonthGenerator(double longitute, double latitude, int month, double[][] drywet, double[][] wetwet)**

// Returns the longest number of consecutive mode (WET or DRY) days in forecast.

**int numberOfWetDryDays (int[] forecast, int mode)**

/\*

\* Analyzes the forecast array and returns the longest number of

\* consecutive mode (which can be WET or DRY) days in forecast.

\*/

**int lengthOfLongestWetDrySpell (int[] forecast, int mode)**

**}**

Use the main method as a driver to test your methods. To generate the weather for location at longitude -98.76 and latitude 26.70 for the month of February do:

**java WeatherGenerator111 -98.76 26.70 3**

public static void main (String[] args) {

int numberOfRows = 4001; // Total number of locations

int numberOfColumns = 14; // Total number of 14 columns in file

// File format: longitude, latitude, 12 months of transition probabilities

// Allocate and populate arrays that hold the transition probabilities

double[][] drywet = new double[numberOfRows][numberOfColumns];

double[][] wetwet = new double[numberOfRows][numberOfColumns];

populateTransitionProbabilitiesArrays(drywet, wetwet, numberOfRows);

/\*\*\* WRITE YOUR CODE BELLOW THIS LINE. DO NOT erase any of the lines above. \*\*\*/

// Read command line inputs

double longitute = Double.parseDouble(args[0]);

double latitude = Double.parseDouble(args[1]);

int month = Integer.parseInt(args[2]);

int[] forecast = oneMonthGenerator(longitute, latitude, month, drywet, wetwet);

int drySpell = lengthOfLongestSpell(forecast, DRY);

int wetSpell = lengthOfLongestSpell(forecast, WET);

StdOut.println("There are " + forecast.length + " days in the forecast for month " + month);

StdOut.println(drySpell + " days of dry spell.");

for ( int i = 0; i < forecast.length; i++ ) {

// This is the ternary operator. (conditional) ? executed if true : executed if false

String weather = (forecast[i] == WET) ? "Wet" : "Dry";

StdOut.println("Day " + (i+1) + " is forecasted to be " + weather);

}

}