CS 2334: Programming Structures and Abstraction

Project 1

String and Date/Time

Due Date: Sep 23, 2020

Fall 2020

100 pts

1. Objective:

The objective of this programming project is to implement a Java program which can read a file, parse data, perform some calculation on data, and some basics on Date/Time. After completing the project, students will have an intermediate understanding of reading data from a file and performing some calculation on data and basics of Date/Time function in Java.

2. Project Specification:

2.1. Overall Program Behavior:

In this project you have to perform some operations on Date/Time and String which will be extended in the later projects in this semester. You are also required to read the file provided, Mesonet.txt. You will require some edit/modification on data to get the station ID(s) (STID: four letter code). There are more than 100 stations, therefore try to declare an array with a small size and expand the array whenever needed.

We provided Driver.java and Mesonet.txt [Pls, don't edit these two files]. We also provided DateTimeOne.java, which is incomplete, you have to complete it. you are required to write DateTimeOne.java, HammingDist.java and README.md.

2.2 Input Format:

For the first part, that includes Date/Time, pls look at the Driver.java file where the tasks/outputs have been described. For more help on Date/Time methods, you can look at Oracle website: https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html. For the second part, we have added a text file which contains some data including the station ID (STID) in the first column. we will calculate the Hamming distance between the stations. The definition of Hamming distance has been described in Section 3. The input format is given below for illustration purpose only, the actual code on Zylab will be tested for several input sets which are different than the input given as example. Two station IDs will be entered. For example, NEWK and WEBR: (the format is like below)

```
HammingDist hammDist1 = new HammingDist("NEWK", "WEBR");
System.out.println(hammDist1);
```

2.3 Output Format:

The format of the output is as follows (For Norman, the four-digit code is NRMN):

```
The Hamming distance between Norman and NEWK is 3; between Norman and WEBR is 4.
For NEWK: Number of stations with Hamming Distance 3: 20.
```

```
For WEBR: Number of stations with Hamming Distance 4: 90.
```

2.4 Description of output:

First line is showing the Hamming Distances between NRMN (this is fixed here) and NEWK, and between NRMN (this is also fixed here) and WEBR. Here, the result is 3 and 4 respectively.

The second line is for NEWK: You need to calculate the distance of all the stations from NEWK; however, you will print only the number of stations for Hamming distance which you get in the first line. In this case, it is 3.

The third line is for WEBR: You need to calculate the distance of all the stations from WEBR; however, you will print only the number of stations for Hamming distance which you get in the first line. In this case, it is 4.

Pls, print the output in this specific format, since Zylab will grade you automatically.

3. **Hamming Distance: T**he Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other. In a more general context, the Hamming distance is the number of positions that differs between two sequences. The following examples will make the idea clear.

Example: "COME" and "GONE"

C	**O**	**M**	**E**	**Hamming Distance**
G	**O**	**N**	**E**	
1	**0**	**1**	**0**	**1+0+1+0 = 2**

Another example: "GONE" and "NONE"

G	**O**	**N**	**E**	**Hamming Distance**
N	**O**	**N**	**E**	
1	**0**	**0**	**0**	**1+0+0+0 = 1**

For more examples, pls visit: https://en.wikipedia.org/wiki/Hamming_distance

4. File names:

File names for this project are as follows:

We are providing: Driver.java, Mesonet.txt (Don't modify these two files)

We are providing but you will add code: DateTimeOne.java

You will write: *HammingDist.java and* README.md

5. Points Distribution:

Bits and pieces	Points
Automatic grading	60
README.md (will be graded during review)	20
Code Review	20

6. Submission Instructions:

This project requires the submission of *soft copy on ZyLab.*

Plagiarism will not be tolerated under any circumstances. Participating students will be penalized depending on the degree of plagiarism. It includes "No-code" sharing among the students. It can lead to academic misconduct reporting to the authority if identical code is found among the students.

README.md: This documentation should contain understanding of the problem and your problem-solving approach, description of methods and variables used. This file will be reviewed by an instructor along with the codes during code review.

6. Late Penalty:

Submit your project on or before the due date to avoid any late penalty. **A late penalty of 20*% per hour* will be imposed after the *due date/time***. After five hours from the due date/time, submission is not allowed.

Good Luck!!