CMPE 343
Fall 2023
Programming Homework 3

This assignment is due by 24 December 2023, 23:59.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:

   - Recitation 1: 04.12.2023, 19:00-20:00, https://tedu.zoom.us/j/98990267511
   - Recitation 2: 11.12.2023, 19:00-20:00, https://tedu.zoom.us/j/98990267511
   - Recitation 3: 18.12.2023, 19:00-20:00, https://tedu.zoom.us/j/98990267511

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK

In this homework, you must implement your own graph data structure by taking inspiration from your textbook and use it to help to solve problem. You are not allowed to use any external library or .jar file. Any solutions without using graph data structure are not evaluated!

Q1(50 points):

Imagine you established a new electricity distribution company. For each 10 km, you should use one electricity pole and you should use the minimum number of poles that will provide electricity to cities you are responsible for. For this reason, you need to create a kind of map.

The input.txt file given to you contains the name and coordinates (x and y, respectively) of the city. Example input.txt is as follows:

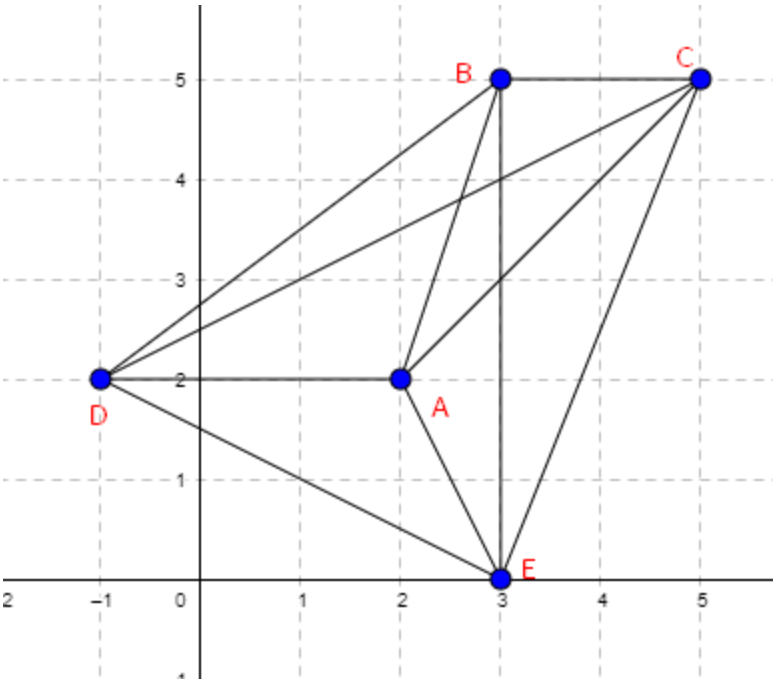
```
A,2,2
B,3,5
C,5,5
D,-1,2
E,3,0
```

You should read input.txt file, and construct your graph based on this information. For example, the first line means that City A is on coordinate x=2 and y=2, or the second line means that City B is on coordinate x=3, y=5.

You can assume that **there is always a path between any two cities** and you should use **Euclidian distance** to calculate length of this path. For example, there is a path from cities D to E and E to D. Their lengths are equal to:

$$\sqrt{(-1-0)^2+(2-3)^2}=\sqrt{2}$$

Example visualization of cities and paths are as follow for input.txt:



Your goal is to find paths to provide electricity to all cities and require the use of the least number of poles. While printing, you should print the path in ascending order of their length. Also, your path starts with city which comes from alphabetically first.

Here is example input and output:

Input.txt	// you should read .txt file name from the user
Paths are:	// you should print shortest path first. And path should start with
B-C: 2.0	// city name which comes from alphabetically first.
A-E: 2.2	
A-D: 3.0	
A-B: 3.2	

Q2(50 points):

You have a map of Turkey, and you want to start from one city and finish your trip in another city. However, there may be cities you want to visit during this trip. Therefore, you have to visit these cities as well. In this question, your task is to find the shortest route from your starting city to your ending city. This route should also include the cities you want to visit. You should read Turkey map from the .txt files which shown in following image:

```
Eskisehir,Ankara,10
Eskisehir,Bolu,20
Eskisehir,Bilecik,5
Eskisehir,Kutahya,30
Eskisehir,Afyonkarahisar,50
Eskisehir,Cankiri,10
Eskisehir,Kastamonu,5
Cankiri,Kastamonu,12
Ankara,Istanbul,14
Istanbul,Cankiri,30
Kastamonu,Corum,14
Corum,Amasya,14
Amasya,Tokat,12
```

You should read this file line by line and construct **an undirected** graph with given weight. For example, the first line means that there is a path between Eskisehir and Ankara with length of 10.

To find a shortest path, first you should read source city, destination city and number of cities which you want to visit, and name of these cities from the console, respectively. If there is more than one city to visit, you should visit them based on console order. For example, you read cities to visit from the console as A, B, C. Then, you should keep this order during your trip and first visit A, then B, then C.

Example inputs/outputs are as follow:

```
input.txt          // you should read input txt file from the user
Eskisehir          // source city
Kastamonu          //destination city
0                  // number of city you want to visit
Routes are:
Eskisehir-Kastamonu
Length of route is: 5
```

```
input.txt          // you should read input txt file from the user
Eskisehir          // source city
Kastamonu          //destination city
1                  // number of city you want to visit
Cankiri            // City to visit
Routes are:
Eskisehir-Cankiri-Kastamonu
Length of route is: 22
```

```
input.txt          // you should read input txt file from the user
Eskisehir          // source city
Kastamonu          //destination city
2                  // number of city you want to visit
Ankara             // first City to visit
Cankiri            // second City to visit
Routes are:
Eskisehir-Ankara-Istanbul-Cankiri-Kastamonu
Length of route is: 66
```

You can find example inputs/outputs on VPL.

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.

- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You should test your Java source files on (if) available Moodle VPL environment to ensure your code solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input. You should pass that task's VPL test case successfully.
- A **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section). Include this report in your zip file and upload it to the PA Report Submission screen in LMS.
- For given task, only code or report submission will not be graded. In other words, you should submit both correct code solution and its related report for the task in order to be graded.

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%2.5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%15): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation, Functionality (%20): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%7.5): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%5): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

GRADING:

- Codes (%50, Q1:25, Q2:25)
  - Available test cases evaluation on VPL: %15, (Q1:7.5, Q2:7.5)
  - Hidden test cases evaluation: %15, (Q1:7.5, Q2:7.5)
  - Approach to the problem: %20, (Q1:10, Q2:10)
- Report (%50)
  - Information: %2.5
  - Problem Statement and Code design: %15
  - Implementation, Functionality: %20
  - Testing: %7.5
  - Final Assessments: %5

Submitting report without codes will not be evaluated!!

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on 24.12.2023.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload your codes to VPL and your report to Submission area in .pdf format.

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-------------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-------------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//-----------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//-----------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TA, Bedrettin Çetinkaya. Thus, you may ask her your homework related questions through HW forum on Moodle course page.