CMPE 343
Fall 2023
Programming Homework 2

This assignment is due by 23:59 on Friday, 15 December 2023.

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: 20.11.2023, 19:00-20:00, https://tedu.zoom.us/j/98990267511
    - Recitation 2: 27.11.2023, 19:00-20:00, https://tedu.zoom.us/j/98990267511
    - Recitation 3: 04.12.2023, 19:00-20:00, https://tedu.zoom.us/j/98990267511
    - Recitation 4: 11.12.2023, 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 directed graph data structure are not evaluated!

Q1(50 points):

In this question, you will implement a directed graph to represent a flight network and perform various operations on it. The flight network consists of city as nodes and flight routes as directed edges. You should read flight network data from the txt file. Also, txt file has the following structures:

```
Ankara,Istanbul
Istanbul,Izmir
Ankara,Eskisehir
Ankara,Amsterdam
Istanbul,Amsterdam
Amsterdam,Paris
Amsterdam,Konya
Konya,Eskisehir
Berlin,Ankara
Paris,Berlin
```

Each line represent flight from source city to destination city, e.g. the first line means that there is a flight from Ankara to Istanbul.

Your purpose is to find all city that can be reached from a given city within a certain number of stops (hops). Then, you should print that city with existing hops in ascending order.

Here is example input and output:

```
Input.txt                    // you should read .txt file name from the user
2                            // you should read the number of hops from the user
Ankara                       //you should read the source city from the user
Number of total routes: 4
Routes are:
Ankara-Amsterdam-Konya
Ankara-Amsterdam-Paris
Ankara-Istanbul-Amsterdam
Ankara-Istanbul-Izmir
```

<u>You can assume that there will be no loops in your graph.</u>

Q2(50 points):

James Bond has a top-secret mission. He must enter a maze and find some treasures. In this homework, you should help him, and you are expected to create a maze solver algorithm. Lucky for you, you have a map of the mazes. These maps contain 3 groups of characters as shown in the example maze in the next page:

- Walls are represented with “+”, “-“, and “|” characters. When you see one of these characters, you cannot go any further.
- Available paths are represented with lower case characters such as “a”, “b”, “c” and etc. When you see one of these characters, you can move forward.
- Treasures are represented with capital “E” characters. When you see this character, this means that you can find a path successfully.

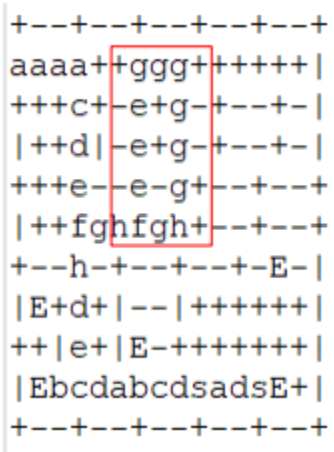
```
+--+--+--+--+--+
aaaaaaaaaaaaaaE|
+b+c+--+--+--+-|
|d++|--+--+--+-|
+e++--+--+--+--+
|fgfg-+--+--+--+
+--+h-+--+--+--|
|a+|d+|--|d+bcE|
+a+|e+|E-+d+a++|
|abcdabcdsadsda|
+--+--+--+--+--+
```

For the above maze, there are 3 “E” characters, and all are reachable. Therefore, you should print all paths according to increasing order of paths lengths. You should read the maze file name from console.

Here is the sample argument: The output for the above maze is as follows. Please check your program with this input as well as the others that you will create. Note that we will use other input when grading your assignments.

```
Maze1.txt        // you should read input txt file from the user

3 treasures are found.

Paths are:

1) aaaaaaaaaaaaaE

2)aabdefgfghdedabcE

3)aabdefgfghdedabcdsadsabcE
```

Also, note that maze can contain a path that has a loop as shown in the following maze with a red box. For this case, your algorithm shouldn't get stuck in the loop. Therefore, you should avoid visiting already visited path.



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 15.12.2023.

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.