**Assessment item 3**

**Assignment 3**

**Value:**20%

**Due date:**20-May-2014

**Return date:**10-Jun-2014

**Submission method options**

EASTS (online)

**Task**

Assessment 2 has total **40** marks. Marks will be scaled according to the value of the assignment.

From the textbook (page 418 of 3rd edition or page 429 of 4th edition) we have seen the maze program. In this assessment, you are required to use a graph class to represent the maze. Then a path should be generated with the entrance and exit as endpoints. Your task is (1) to use a depth-first search to travel through the maze; and (2) to use Dijkstra’s shortest distance and shortest path algorithms as described on pages 732-743 of 3rd edition or pages 759-771 of 4th edition by assuming weight to be 1 for each edge. This could happen if there are more than 1 path connecting starting and ending maze cells.

**Details for Implementation**

1.    We assume a maze is read from a text file. ***A maze file starts with two integers representing the number of rows and the number of columns of the maze cell*.** A maze cell corner is represented by + and walls by “-“ (horizontal walls) and “|” (vertical walls). The below shows an example of maze files which represents a maze with cells in 4 rows and 5 columns. We already consider the left-top cell as the starting cell and the bottom-right cell as the ending cell. They are marked by “s” and “e” respectively in the above picture. However both markers are not in a maze file. Please note in this representation, a cell with all four walls is written as

+-+

| |

+-+

If any of “-“ or “|” is missing, there is no wall on that side. The corner “+” is always there.

2. Your program must display a representation of the maze on screen as it is in a maze file. Since graphics is not a requirement for this subject, you can use the above text-based representation for this if you don't want to delve into Java's graphic facilities. After your algorithm has found out path from the start and the end, you program shall print the path by writing “v”, “^”, “<” and “>”. For the above maze, we know a path which can be printed as

3. One of important steps in your program is to convert the text-based maze representation into a Graph object. Here is the way to represent a maze by a graph. Each cell becomes a graph vertex and there is no wall between two neighbouring cell, then there will be an edge in the graph representation. Please note that the corners “+” can be ignored. Actually corner marks are added for better visual effect. For the above maze, the graph looks like

4. The ordinary Graph class is implemented by Michael Main which can be downloaded from his website or a copy of which can be obtained from the subject site on the Interact. In this class, the depth-first search algorithm has been implemented, including the facility to extract a path. However you may need to modify it to suit your case.

5. To use Dijkstra’s algorithm for the shorted path, please implement a new class for graphs with weighted edges. Either you use the ordinary Graph class as a superclass for your implementation, or you start it from scratch by following the pattern used in the ordinary Graph class.  
Provide two extra methods to implement Dijkstra’s shortest distance and shortest path algorithms as described on pages 732-743 of 3rd edition or pages 759-771 of 4th edition. Please carefully read the algorithm explanation and examples in those pages so that you fully understand the whole algorithms. The shortest path algorithms only differ from the base shortest distance algorithm in keeping a record of previous vertex for each vertex. It would be helpful to write a method for printing a path from the start vertex to a given vertex according to the required output.

6. You can test your program on two provided maze files maze01.mz and maze02.mz. They can be downloaded from the Interact site.

**Rationale**

This assessment has been designed to test and demonstrate the ability to write Java programs for complex data structure concepts.

**Marking criteria**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Assessment 3 (Total marks 40)** | | | | | |
| Criteria | Marks | Pass | CR | DI | HD |
| **a.       Execution** | 5 | Programs execute without trouble (5). 2.5 marks will be deducted for each program that fails. | | | |
| **b.    Design & implementation** | 20 | depth-first search algorithm (8), Dijkstra’s shortest distance (6), Dijkstra’s shortest path algorithm (6). To get "Pass" coding with minor errors required and to get "HD" all codes must be correct, professional with appropriate comments. | | | |
| **c.     Presentation** | 5 | Few comments and minor indentation errors (2.5) | Proper indentation in each code block and comments in major lines and blocks (5) | | |
| **d.    Submission** | 10 | Provide necessary files (5), Discussion on algorithm, instruction on compiling and running your program (5). | | | |

**Presentation**

o programming solutions to the tasks;  
o javadoc documentation of your classes;  
o any input files required to run your programs and examples of output files; and  
o a discussion of important algorithms used and how your programs should be compiled and run.

Please submit your project through EASTS system.