COP3503 Computer Science 2
Dr. Andrew Steinberg
Spring 2023

Programming Assignment 2
Magic Maze
Due: 2/19/2023 at 11:59pm

Objective: Students will apply concepts of advanced Backtracking.

Your solution for each test case must run within 1 second. Otherwise, no credit will be given for that test case!!!!

Assignment Description: Kenny is with his friends at the county fair. They got tickets to enter in the magical corn maze. The maze consists of teleportation pads, that allow you jump through various spots in the maze. Kenny is really scared to go in, but his friends promise they will always be together in the corn maze. Unfortunately, they got separated and now Kenny is alone in the corn maze. Kenny is freaking out until he gets a text from his friend that they are waiting for him to get to the end. They promise to guide Kenny to the end. However, his phone has a low battery, so he must hurry (or else he is trap there). Kenny's friends remind him that there are some magical teleportation pads within the maze that will help him get out fast before his phone battery dies. However, there are some that can send Kenny to dead ends. Your goal is to get Kenny out of the magical maze before his phone dies (hence the running time constraint restriction).

The Maze Files and Understanding the Maze Itself:

You are provided a total of 5 text files. Inside each text file is a maze. Here is a sample below.

```
*******@3*****4
*******@@@@@@@@
*******@******5
2*****3@4***@@@
@@@@@@@@@@@@@@@
*******@*******
*******1***2**@
*******@******@
*******@@@@@@@@
*******@5@***1@
*******@X@****@
```

- The number of rows and columns are all different in each text file, please do not assume they are the same. However, I can assure you that the time limit will remain the same for the grading. Meaning, the biggest maze provided will be the same size for when the graders test your code.
- The '*' represents a valid space that Kenny can walk on.
- The '@' represents an invalid space that Kenny cannot walk on. Think of it as an obstacle.

- The 'X' represents the exit square. This is basically the destination that Kenny needs to get to be out of the magical maze.
- Kenny can only walk in 4 directions. Up, down, left, and right. Diagonal moves are not allowed.
- The digits such as '1', '2', '3', etc… represent magical squares. These squares will teleport you to other matching digit in the maze. For example, the '1' in the above figure, if walked on, will transport you to the other '1'. It will **NOT** transport you to the different digit spots.
- Kenny will always start at the bottom left corner of the maze. That is the entry point technically.

Implementation Details for this Assignment:

1. You are going to create a Class called MagicMaze.
2. The MagicMaze Class must have the following attributes:
    a. 2D primitive char array that will store the entire input maze from the provided text file.
    b. A String object that stores the maze number (maze1, maze2, etc…)
    c. 2 primitive integers that each hold the number of rows and columns in the 2D maze
    d. You are welcome to add additional attributes that feel may be necessary.
3. The MagicMaze Class has one overloaded constructor with three parameters.
    a. A String Object that represents the name of the maze text file
    b. A primitive int that represents the number of rows.
    c. A primitive int that represents the number of columns.
4. The MagicMaze Class must have some method that will perform scanning the maze from the provided text file into the respective class attribute. This is like Programming Assignment 1.
5. The MagicMaze Class has a non-static method called solveMagicMaze, which will be the actual problem solving (getting Kenny out of the maze) that you will need to implement. The method returns a Boolean value. If the value is true, then the maze was solved successfully, otherwise the maze was not solved successfully. You must use Backtracking to solve this maze. I would recommend creating a recursive method like the N-Queens problem we did in class.
6. You are given 5 test mazes inside the provided text files. All text files are in the same directory as the driver and solution file you are going to implement. DO NOT CREATE SUBFOLDERS! Points will be deducted and will not be given back!
7. You must use Backtracking in your implementation. If Backtracking is not used, then an automatic 0 will be given in this assignment as the overall grade with no partial credit.
8. **Do NOT use any Graph Algorithms in this assignment. We have not covered graph algorithms yet! Any Graph Algorithms used will result in automatic 0 in the assignment as the overall grade with no partial credit.**

9. You are welcome to create helper methods in your solution class, but please do not modify the driver class in any way. This will result in your code not working properly when grading, which will result in points being deducted that are not going to be given back.

The Provided Driver File: A driver file (`MagicMazeDriver.java`) has been provided for you to show you how the methods are called along with a test case based on maintaining the given maze. In this assignment, you will notice the main method uses the args parameter. In order to test your file when running your code you will need to type some extra things into the terminal. Besides typing java and the name of the class you want to run, you will need to include text file to be read, the number of rows, and the number of columns. For example, `java MagicMazeDriver maze4.txt 15 20` will run the driver class along with sending the respective text file to be solved, the number of rows in that particular maze, and number of columns as well. **Please note that you can only test one maze at a time. The provided python script will test all 5 mazes at once to ensure they are within the time constraint requirements.**

What to submit: Submit a file called `MagicMaze.java` to webcourses. You are not required to submit the driver file as that will be provided for the graders to test your code. Please make sure the driver file provided works for your code. Any name changes may cause your program not to work when graded, which will result in a lower score on the assignment and would not be changed.

Important Note for running Eustis: Many of you are probably using IDEs like Netbeans and Eclipse to build your Java Solutions. Please note that some of these IDEs will automatically place your Java file in some sort of package. Please make sure your Java file is not defined in some package as this can result package private errors. Any such error that occurs during the grading will not be fixed and points will be deducted as such in accordance with the respective categories in the rubric. Also, DO NOT create a main method in your solution file!! This will result in your code not running properly with the runner file which will result in points being deducted from the respective categories.