COP 3503C Programming Assignment # 4

BFS

Time Limit: 4 seconds (per input case)

Read all the pages before starting to write your code

What should you submit?

Write all the code in a single Main.java file. **Submit your** Main.java file**.**

Please include the following commented lines in the beginning of your code to declare your authorship of the code:

/* COP 3503C Assignment 4

This program is written by: Your Full Name */

Compliance with Rules: UCF Golden rules apply towards this assignment and submission. Assignment rules mentioned in syllabus, are also applied in this submission. The TA and Instructor can call any students for explaining any part of the code in order to better assess your authorship and for further clarification if needed.

Caution!!!

Sharing this assignment description (fully or partly) as well as your code (fully or partly) to anyone/anywhere is a violation of the policy. I may report to office of student conduct and an investigation can easily trace the student who shared/posted it. Also, getting a part of code from anywhere will be considered as cheating.

Deadline:

See the deadline in Mimir. The assignment will accept late submission up to 24 hours after the due date time with 10% penalty. After that the assignment submission will be locked. An assignment submitted by email will not be graded and such emails will not be replied according to the course policy.

What to do if you need clarification on the problem?

I will create a discussion thread in webcourses and I highly encourage you to ask your question in the discussion board. Maybe many students might have same question like you. Also, other students can reply and you might get your answer faster. Also, you can write an email to the TAs and put the course teacher in the cc for clarification on the requirements.

How to get help if you are stuck?

According to the course policy, all the helps should be taken during office hours. Occasionally, we might reply in email.

Problem Description: Tricky Maze

Your friend Gustavo is hopelessly lost in a maze, and you would like to help him get out. Luckily, his cell phone is fully charged, and you can call him and give him directions to follow. He's spent so much time lost in the maze, he insists that you get him out as fast as possible.

The maze can be modeled by a two-dimensional grid with *r* rows and *c* columns. Gustavo's location is labeled with the character '*' and the one location that allows for escape out of the maze is labeled with the character '$'. In most circumstances, Gustavo can move to the left or right by one square on a single row, or move one square up or down in a single column. However, there are some exceptions to this rule. Some squares in the grid are forbidden. These are marked with the character '!'. Other squares are teleportation squares. Each of these are marked with a capital letter. If Gustavo is on a square with a letter, say 'A', he can teleport, in one move, to all other squares labeled with the letter 'A'. Same goes for all of the other capital letters. ***Note that from a teleportation square, one can always choose not to use the teleportation feature and can still move left, right, up or down by one square. Thus, one can travel from a square labeled 'A' to an adjacent square labeled 'B', or an adjacent square labeled '.'.***

The Problem:

Given the size and contents of the maze, figure out the fewest number of moves Gustavo needs to make to get out of the maze.

The Input (must be standard input (no file i/o)):

The first line of input contains two space separated integers, $\boldsymbol{r}$ ($2 \leq \boldsymbol{r} \leq 1000$) and $\boldsymbol{c}$ ($2 \leq \boldsymbol{c} \leq 1000$), representing the number of rows and number of columns in the grid, respectively.

The following $\boldsymbol{r}$ lines contain $\boldsymbol{c}$ characters each. The i^{th} line of these lines contains the contents of the i^{th} row of the grid, from left to right.

It is guaranteed that exactly one of the grid characters will be '*' and exactly one of the grid characters will be '$'. All grid characters that represent regular squares will be labeled with the character '.'. All forbidden squares will be represented with the grid character '!'. All other squares will be capital letters, representing various teleportation squares. If a letter appears in the grid, then it will appear in at least two separate grid squares.

Partial Credit Input Restrictions:

In some of the input cases (enough to allow a maximum score of 70%), no teleportation squares will exist.

In some of the input cases (enough to allow a maximum score of 90%), no letter will appear in the grid more than 10 times.

In the last set of input cases worth 10% of the grade (to raise your grade from 90% to 100% max), there are no restrictions on the number of times an individual letter appears in the grid (beyond the size of the grid and the other grid square requirements).

The Output (standard console output):

If Gustavo can get out of the maze, output a single integer representing the fewest number of moves it will take him to get out. If he can't get out, output "Stuck, we need help!".

Restrictions:
You must have to use the BFS strategy in your solution.

Sample Input **Sample Output**

3 4 .$!* .!!.	8
4 2 .. *! !$ ..	Stuck, we need help!
6 6 C..$B. ..... !!!!!! ....CB .*.AAA C.B.CB	4

What To Submit

For this assignment, please submit a single Java program named **Main.java**

Hints:

1. The BFS concept, code and then the ElevatorTrouble problem discussed in the class/recording would be your initial starting point to think about the solution
2. Then the lab problem Knights jump should help you significantly.
3. Based on the above three concepts and codes, try to solve the problem for the first partial credit restriction.
4. Now, if you would like to enhance your code for 100% credit, consider storing all the locations for each letter in a suitable data structure. During the BFS process, you can simply consider that all the locations of the same letters are connected.
5. You may need some other logic and variables and list to decide whether you will enqueue the location of a letter or not during the BFS process..

Good Luck!