Maze Solver

# Overview

Your program is to read a text file that has the schematics for a maze in the form of a graph adjacency list. Using this list, your program will determine the shortest path out of the maze.

# Program Inputs

At startup, your application must prompt for the location of a file that has the adjacency list of an undirected graph. The first line of the list contains the names of all the nodes. The second line contains two nodes, indicating the start and end of the maze (in that order). Each additional line contains a node in the graph followed by each node to which it is adjacent. In all cases, the nodes are comma-separated. The input file may contain multiple mazes, each separated by a blank line. It is possible for a maze to have dead ends. It is possible for a maze to have cycles. It is possible for a maze to have multiple acceptable solutions, but it is also possible that a maze cannot be solved. An example of this input is provided below.

# Algorithms and Outputs

Using standard graph traversal algorithms (pick your favorite), discover the SHORTEST path through each maze. Display the path by outputting the sequence of nodes to the UI. If more than one path ties for shortest, choose one path to display. If a maze cannot be solved (i.e. the end node cannot be reached from the starting node), your output should simply indicate this with “Maze cannot be solved”.

**Example**

|  |  |
| --- | --- |
| **File Input:** A,B,C,D,E,F A,E A,B B,A,C,D,F C,B D,B,E,F E,D,F F,B,D,E  **Expected Output:** Maze 1 Solution: ABDE |  |

The image on the right is a drawing of the example graph (you do not need to output that!) Output would continue for any other graphs found in the file.

# Rubric

**Automatic Zero:** Doesn’t prompt the user for a text file for data entry.

(10 points) Correctly parses text document contents

(40 points) Correctly identifies solutions for every maze listed in the file