It currently works as intended, with the capability to read cave layouts from a file or directly from a string. The program uses a depth-first search algorithm to find the path.

The program was tested with various cave layouts, including some that required backtracking. Tests covered edge cases such as no valid path, multiple possible paths, and out-of-bound movements. These tests helped identify and fix issues in the code, ensuring the algorithm worked correctly.

I learned how to implement a depth-first search algorithm to solve a pathfinding problem using a stack and how to manage different input sources for initializing the cave layout.

I enjoyed the problem-solving aspect of the project, particularly devising an algorithm to find the unique path in a 2D grid-based cave.

initial handling of different input sources (file and string) was a bit confusing. It could have been clearer if there were separate constructors for each input type right from the start.

If given more time, I would like to add more functionality to the project, such as handling multiple start and end points, and possibly implementing other pathfinding algorithms like A\* or Dijkstra's algorithm to compare their performance. Additionally, it would be interesting to create a graphical representation of the cave and the pathfinding process.