1) Discuss the structure, behavior, and practical uses of the two data structures (grid and stack) used in our maze solving program.

The grid structure is implemented in the getMaze() function of the program, which seeks to read the maze that is contained inside of the text file and transfer the contents of that file to a graph so that Python is able to manipulate the maze. The stack structure is used to solve the maze, where the program checks the different parts of the code starting with the upper level. Once the program has checked whether or not there is a '-', or '#' in the program, the position is pushed into the stack so that each point will not be visited more than one time.

2) Discuss the space and time efficiencies of the stack-based backtracking algorithm used in our maze solving program.

The efficiency of the space and time of the stack's backtracking algorithm is the optimal solution for this situation because the algorithm takes the approach of a brute force search but is probably not the best in time efficiency. Despite this drawback, I still believe that it is the best for this situation because it "does not leave any stone unturned" in the way that it checks every spot before moving on to the next one.

3) Discuss how you could use a graph data structure to represent and solve a maze.

A graph data structure can be used to solve a maze in the way that the nodes are the different spots of the maze and the edges are the way the program navigates through the maze. In Python, the graph would be used to store the data from the maze and make it readable by the Python program. When this has been accomplished, we can use another graph structure to store the spaces that are visited and move on to the next spot.