Project #5
CS 2210 – Fall 2022
Christopher LaFave

I. Requirements: Create a program that uses a queue and a breath-first search to solve a maze.

II. Design: I had a 2D array that mapped out the maze and a queue to store which location needed to be searched next. I had a Location class, a Coord class, and a Search method to aid with the mapping out and implementation of the code.

III. Security Analysis: I was told I didn't need to handle bad input, so there is no error handling for any unexpected input, which throws a huge security risk. Other than that, I sanitized the inputs relatively well, so I don't think there would be a big security risk otherwise.

IV. Implementation: While initializing the maze, if I run across the starting point I note where it is. I start the search at the starting point, and I put all adjacent locations into the queue, unless they are out of bounds or already checked. I then continue to push those positions off of the queue until we reach the target point. If the queue runs out and we haven't reached the target point, that means that there is no solution to the maze.
To print the solution path, I first store the distance at the target point, and then I push all of the target point's previous locations onto a stack. I then just empty out the stack by printing all of those locations.

V. Testing: I only tested was the Gradel mazes.

VI. Summary/Conclusion: My program runs properly and gets all the correct outputs on Gradel.

VII. Lessons Learned: I learned how to use a queue to perform a depth first search, became more comfortable organizing Java code, and learned how to do battle with Gradel.