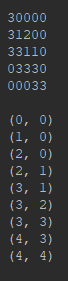
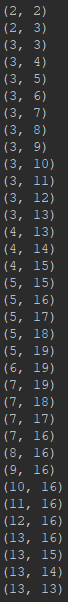
## Goal

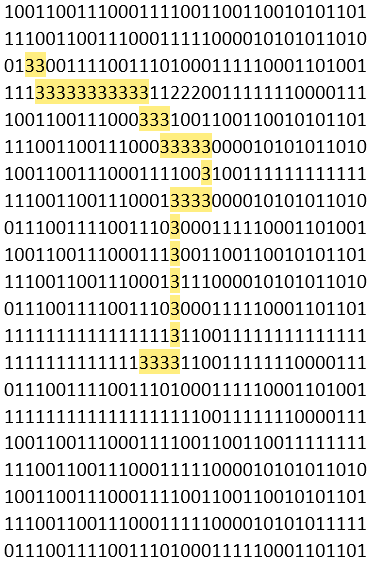
The goal of this assignment was to use the code base given for the maze, and have the program figure out a path through the maze if the maze was solvable.

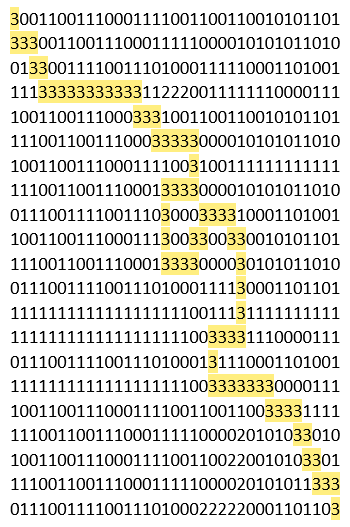
## Testing

For this project I did a mix of unit testing, and input/output testing to verify that my code worked well. I unit tested each method in the classes I created to ensure that they performed how I was expecting them to perform. For each method there was at least 1 test method created. Some methods had multiple test methods that would touch on some edge cases, such as thrown exceptions or when critical information was missing. In the unit testing I would simulate a situation where you would want to use the method in question, then I would assert key values to ensure that my methods were running correctly. Unfortunately, I had to rely on other methods to setup the simulations correctly, which typically I do not want to do to make sure I am testing a single bit of functionality at a time. I did not test constructors, or the Position class (I just added a single extra constructor to it). There was a couple of methods inside the Maze class that I did not test either, these methods were copied methods that are already established as working, with a minor change in a single value (specifically setting or checking the value of the current Position in the grid).

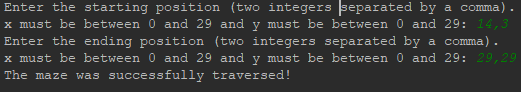
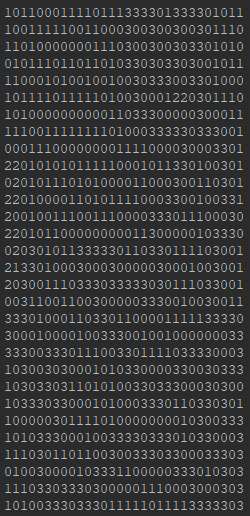
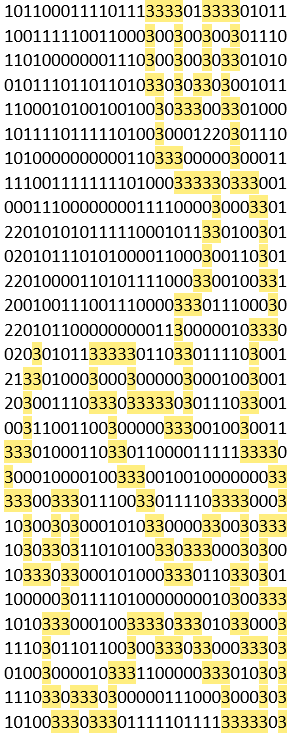
 My input and output testing was how I tested the new methods in the MazeSolver class. I started using “testfile.txt,” to make sure I got the simplest of mazes solved before trying the larger ones. Once I realized that the larger mazes gave me a stack overflow, I decided to modify “testfile.txt” to include more potential paths. To the side is the final solution for testfile.txt.

 The next step was to get the solution working using “testfile3.txt.” Below to the left is the original solution for “testfile3.txt” that was copied out of the console and pasted into word, then highlighted all the 3s to easily see the path. Below center and right are “testfile3.txt” solution using custom coordinates for start and end with the far-right picture being the coordinate list. For brevity I did not include every test with custom coordinates.





Lastly for this project I needed something special to try so I had confidence that this wasn’t a coincidence. I copied the maze array from <https://github.com/oppenheimj/maze-generator> and imported it into “testfile4.txt.” I figured this would be the least bias option and would ensure I had a solid maze solver. Below are the results using custom coordinates.



Solution in word on the left

Console output of the solution

on the right.

Coordinates below

Coordinate output 