Throughout the process of completing this problem set, I went through multiple iterations of my solving algorithm. The first, which was my favourite to program, was a maze-solving algorithm which stuck to the right hand side of the maze until it found the exit, which is a guaranteed (albeit slow) method of calculating a solution. The next was a random guess-and-check, which seemed better on average than the stick to the right strategy.

While I was unable to bugfix in time to get accurate results for my testing, I would like to breakdown the strategies, efficiencies, and inefficiencies of the functions.

Function A: Random guess
While this has a worst case scenario of O(infinity), I added a check method which avoids previous states, thus allowing a maximum of n guesses, with n representing the n * n dimensional grid of the maze (or tile). This also results in a best case scenario of O(n), and an average case of O((n+1)! based on random algorithms presented in previous programming courses I've taken.

Function B: Sticking to the right wall
This was a function I envisioned from years of casual maze-solving. The old mantra: if you're in the classical labyrinth, always stick to the right wall, and you'll get out eventually