The most recent version of this project is hosted on GitHub: https://github.com/nylesb/maze-solver.

Here is a description of the files you will see contained in the folder:

- maze-solver/
  - unittests/
    - maze-for-unittests.txt: This is a sample maze that the tests file uses for its unit testing.
    - tests.lisp: This file contains all my unit tests. Its setup to be relatively easy and effective to use and make new tests, but has limitations.
  - .gitignore: The ignore file for Git version control.
  - prog.lisp: This is the actual program.
  - readme.pdf: This file
  - readme.dox: This file as a Word Document
  - test-prog.lisp: The driver for the program. This calls my unittest program, provides examples on how to call my functions and provides a dribble of their output.
  - Breecher_hw1.out: This is the dribble from one run of test-prog.lisp

First, thorough documentation has been included in all source files, so refer to them for specific questions. This is a high level document about the project. My algorithm for this project is rather simple. The navigator from a position tries moving in all directions. Whenever the navigator is on a new position, it tries again to move in each direction. These calls happen recursively. The navigator also keeps track of the current path it's been on, so it won't double back on its path. This is important because for this project no maze is only solvable by crossing over its current path. Note that this concept is slightly different from backtracking. Backtracking occurs when the navigator has exhausted all directions from its current location and hasn't found the solution yet.

Additionally, to put this project together I built a unittesting structure from scratch. Documentation for its use and helper functions are included in that file, if you desire to look at it. As long as everything runs it will tell you expected and actual values, but won't help give specifics of the errors. This is one place the program could be improved.

In terms of the scope of the project, my program fits all the requirements of the assignment. I tried to follow proper formatting and Lisp styles as I could, but this being my first time using the language I'm sure there are things to improve. Solve-maze and create-problem are properly using blocks to create closures around anything they create. My tests use some global variables/functions however. Additionally, there are some cool features that one can apply to solve-maze and create-problem that are worth mentioning.

Solve-maze program checks for invalid starting locations and gives this message if this happens. Solve-maze can be called by giving it a file input containing a maze and a list of starting coordinates, as per the problem assignment. You can also manually give it a maze list and one pair of starting coordinates if you would like too though! Just make sure to use :file or :maze-list when calling the function (refer to documentation).

Similarly, create-problem has some options on its parameters. You can call it without anything to generate a random size from 3-7 maze and it will run three starting positions on that maze. If you'd like, you can specify the size of maze you'd like to generate. To make the random maze it adds walls into a blank maze based on a wall-density parameter, which can also be passed to the function. Lastly, you can

give how many starting coordinates on that maze you would like to run. Unfortunately I didn't know how to test the randomness of this function, so I could not write an automated test for it.