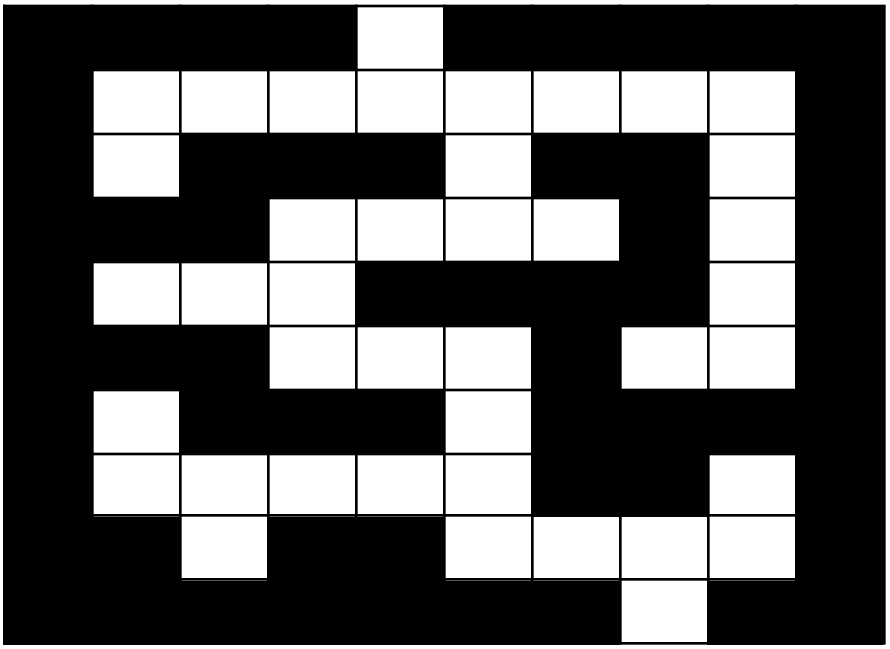
Background

You are going to be building a maze solver for simple mazes such as the following. Don't get started yet -- make sure to read the background and the user stories first.



We expect solutions to be submitted as a link to a private version control repository demonstrating your development history (GitHub, GitLab, or equivalent). Your project should include a README with instructions on how to build/run your solution.

Don't worry about building the perfect solution. (Gold-plating is not a plus.) Instead, your solution will be graded on the following criteria *in this order*:

- Demonstrated approach to testing and validation best practices (unit, integration, end-to-end, etc)
- Software craftsmanship including, but not limited to, clean code practices, SOLID principles, KISS, refactoring, and maintainability
- Solving each user story incrementally instead of attempting to solve the whole problem in one go (preference is 1+ commit per user story)
- Demonstration of problem solving approach, for example, through commit comments
- Algorithmic approach

Path software engineers do most of their work in C++, but you should feel free to use your preferred language. You will be expected to demonstrate good design principles and follow the -isms of whatever language you select.

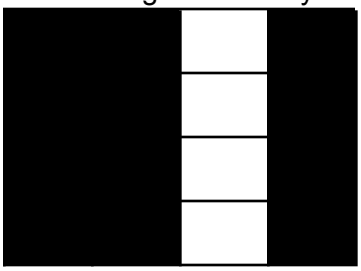
Part 1: Coding

User Story 1

Find the “empty” space in a single-row input such as the following:

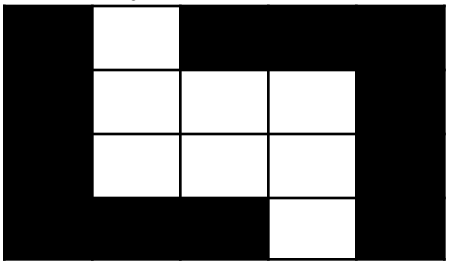
User Story 2

Walk through a “hallway” maze such as:



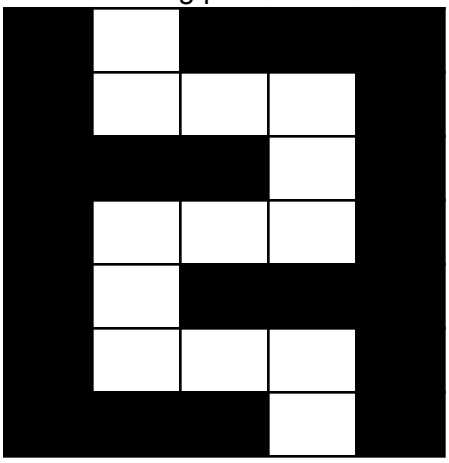
User Story 3

Find a way into and out of rooms such as:



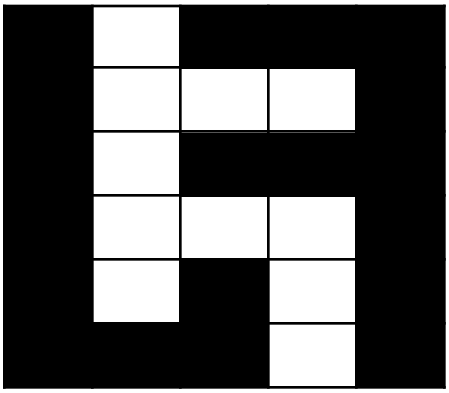
User Story 4

Follow winding paths:



User Story 5

Reach the end of the maze, even if there are dead ends:



Part 2: Reflection/Analysis

Respond to the following prompts in your README. You do not need to update your code any further at this time.

Analysis Story 1

At this point, you may have created a general-purpose solver. If not, try to identify some types of mazes that your algorithm would not solve correctly.

Analysis Story 2

If you kept things simple, it is likely that your algorithm may not be as efficient as possible. Describe the solution's complexity and approaches that could be used to optimize it further.

Analysis Story 3

Moving robots isn't as simple as moving a 1x1 pixel through a maze. Instead, we must plan a path while avoiding obstacles using a collision model. We can approximate this by plotting a path for a 1x3 "ship" through a maze. In addition to moving "backward" and "forward" the ship can also rotate around its center of gravity provided it is in the center of a 3x3

Do not code a solution, but instead describe an approach for decomposing this problem into incremental stories as done for the maze problem above.