CMPUT 174 Lab 8: Dungeon

Theme of this lab: text adventures!

Do you remember how Sheldon Cooper played a text-based adventure game in one of the episodes of "The Big Bang Theory"?

Well, not just Sheldon! **At least** one of your instructors spent quite some time playing a multi-user dungeon game.

> Did you know that even though it's 2023, you can still play a real **multi-user** text adventure game? If you're on Linux (or WSL2), just type in the terminal:
>
> ```
> telnet aardwolf.org 4000
> ```
>
> For Windows and macOS, you can use a client like Mudlet (server name would be **aardwolf.org**, port number is **4000**).

Now, **you** will write a game that is similar to the one Sheldon (and your instructor) played. It will be a simple version, of course. But feel free to enhance it when CMPUT 174 is over!

In this game, a player will be able to move around the dungeon. You worked on a different dungeon game in class so you can reuse some concepts and ideas. However, this game will have a slightly different twist.

First, your game will have more commands.

Second, it will work with any map. You will read the map from a text file.

We created a few maps for you (see below in the **Resources** section). You can use them or create your own.

Learning Outcomes

- Use nested lists to operate multi-dimensional data
- Employ user-defined functions to decompose computational problems
- Identify appropriate parameters and return values for user-defined functions
- Use file input-output to read data from disk
- Use unit tests to ensure correctness of a program
- Apply evolutionary prototyping to design programs step by step

Software Quality Requirements

For this lab, you must apply **all** software quality requirements, **except** Section 7 (User-defined Classes).

Tasks

The following tasks are versions of the same problem. We're learning to write code in an incremental way. Start with the simplest version, and then add more functionalities to make your code more complex. Please do the tasks in order, starting with the first one.

1. Version 1: Minimal version
2. Version 2: Looking around
3. Version 3: Exploring
4. Version 4: Final version

Reflection Questions

Once you're done coding, use these questions to think about your code. It's an essential part of learning because we can never write good code if we don't think about the problem and consider different ways of solving it.

When you demo your lab, a TA may ask some of these questions.

1. You tested your program with all four maps? Try to create your own and share it on Ed Discussion!

2. Our maps have an empty line in the end. However, some maps might not have an empty line. Would your code support such text files?

3. Examine your `look_around()` function. It should call the `is_inside_grid()` function, so your code is likely relatively concise and readable. What if you did not have the is_inside_grid() function? How much longer would your `look_around()` function be?

4. Now, examine your `move()` function. If it calls `look_around()`, it should be pretty concise. What would it look like if it didn't call `look_around()`? How would it violate the DRY (Don't Repeat Yourself) principle?

Resources

Use the following maps to test your code:

- Cave
- Midgaard
- Ghost Town
- Arcadia

Your code must work with any map that satisfies the requirements. During the demo, your TA may use a different map (not one of these four).

Marking

The are no part marks, no in-between marks

4/4	Your code clearly meets all requirements of **Version 4** and all software quality requirements. You clearly understand your code and your answers are correct.
3/4	One of the following: a) Your code meets all requirements of **Version 3** and all related software quality requirements. You clearly understand your code and your answers are correct. b) Your code meets most **Version 4** requirements and most software quality requirements; it runs and does what is expected. However, some minor requirements are missing, or some details in your answers are missing or incorrect.

2/4	One of the following: a) Your code meets all requirements of **Version 2** and all related software quality requirements. You clearly understand your code and your answers are correct. b) Your code meets most **Version 3** requirements and most software quality requirements; it runs and does what is expected. However, some minor requirements are missing, or some details in your answers are missing or incorrect.
1/4	One of the following: a) Your code meets all requirements of **Version 1** and all related software quality requirements. You clearly understand your code and your answers are correct. b) Your code meets most **Version 2** requirements and most software quality requirements; it runs and does what is expected. However, some minor requirements are missing, or some details in your answers are missing or incorrect. c) You put effort into your lab assignment, but your code doesn't run at all or runs with major problems. Missing major requirements, or your answers are mainly incorrect.
0/4	One of the following: a) Incomplete, or very insufficient code, or no submission. b) Code submitted but no show, or no answers, or irrelevant answers.