CMPS 6100 Lab 02

In this lab, you will implement a game!

Unlike Lab 01, this lab does not have automated tests. Your grade for this lab will be based on your implementation.

Refer back to the README.md for instruction on git and how to submit properly to get all the points you've earned.

The Ghost Game (11 pts)

Implement the Ghost Game that we discussed in lecture. In this game, the player is trying to escape a haunted mansion before being captured by a ghost. In order to escape, the player must discover the room containing a portkey which allows them to teleport out of the mansion.

For every turn of the game, the player is presented with a description of the room they are in along with where its exits are. The player then chooses which exit to go through to enter a new room. If the player enters the room with the portkey, they win. If they enter the room with the ghost, they lose.

At the beginning of the game, the locations of the ghost and portkey are randomly decided among all rooms that are not the starting location which is always room 0.

Every turn, the ghost will move from its room into an adjacent room by random choice.

Input

Your game should read in an input file specifying the layout of the mansion. The format of the file is as follows:

- Any lines that begin with a `#` are comments and should be ignored when the file is read in.
- The first non-comment line of the file is the number n, the number of rooms in the mansion.
- The next n lines contain three values separated by colons: a room identifier given as an int, the room name, then a description for that room.
- All lines after that give the connections between the rooms. Each of these lines contain ints separated by colons. The first int is a room ID, the next four values are the IDs of the rooms through its exits. The exits are given by cardinal direction in the order of East, North, West, then South. Any direction which does not have an exit is given a value of -1.

An example input file is provided for you: `the-stilts.txt`

Before running your game on this map, draw it out by hand so that you can refer your map to verify that your game behaves as expected. It may also be easier to create a simpler map to test your game on.

Advice

For this program, I recommend that you start by deciding how you will represent the data for the room layout. After that, implement your game by breaking it up into functions. This will make it easier to develop, test, and debug. Speaking of which, test early and often!

Customizing the game

Please do create your own maps. Create at least one map of your own design.

13. Add the text file for your map and a drawn image of it to your repository. Note the names of these files in `answers.md`. Pro-move: embed an image of the map into `answers.md`.

Also, feel free to change the context of the game. Maybe its a zombie in a hospital, or a vampire, or a tear in space-time that is wandering the halls of a lab. Have fun with it!

Bonus

Remembering Rooms (3 points)

By default, when the player enters a new room, they won't know what is on the other side of each exit. By default, they should be told where the exits are but not what is on the other side.

After the player has discovered a room, remember this information and update the exit messages appropriately.

For example:

```
You have entered the Dining Room...
There is an exit to the East.
There is an exit to the South.

...
User chooses to go East.
...

You have entered the Kitchen...
There is an exit to the North.
There is an exit to the Dining Room to the West.
```

Additional Features (up to 5 bonus points) Other functionality and added features are also welcome! Again, be creative and have fun.

14. If you add new features to the game, document them in `answers.md`.

Epilogue

You've implemented a fun game, continuing to practice and hone your skills as a programmer. I look forward to playing it!