Nano Adventure

The Exercise

In this programming exercise, we'll create the world's tiniest adventure game. In the language & development environment of your choice–Python or JavaScript would be ideal but use what you're most comfortable with–write a program that implements the adventure game described below. Include a small set of unit tests to make sure your code does what is expected. The goal of the exercise is not to demonstrate language or algorithm mastery! Rather, it is to see how you like to explore and understand the problem, and how you translate your ideas into code which solves it. Don't overthink the problem or overcomplicate the solution, just focus on writing simple, expressive code.

The Game

The game will consist of two parts:

- A configuration language that describes the rooms in the adventure game and how they are connected. The game will use this information to tell the user what happens when they move or take other actions in the game.
- An environment in which a user can enter text commands to move around and perform actions in the game. This will be much like a REPL (Read, Eval, Print Loop) that many languages provide. The game will prompt the user for a command, read it, evaluate the command to determine what happens next, and print that result so the user can see it. Then loop to get another command from the user!

The Game Language

The descriptions of the rooms will consist of one or more 'entries' for each room, with each room entry having these attributes:

- id: a number uniquely identifying the room
- description: a textual description of the room

For example, here we describe two rooms in pseudocode:

```
rooms:
  1, "You stand just outside the entrance of a hobbit hole."
  2, "You are in the entrance hallway of the hobbit hole. A wizened
     wizard and small hobbit stand nearby looking curiously at you."
```

In addition, each room has one or more connections to other rooms. We describe each room's connections with attributes:

- roomid: The room id whose connections are being described
- up: The room id reached if the user goes 'up' when in this room
- down: ... 'down' ...
- north: ... 'north' ...
- south: ... 'south' ...
- east: ... 'east' ...
- west: ... 'west' ...

For example, the following pseudocode describes the connections between the example rooms above. Moving west from outside the entrance takes you inside, and moving east from inside (in the entrance hallway) takes you back outside.

```
connections:
  1,0,0,0,0,0,2
  2,0,0,0,0,1,0
```

In this example, each line is in comma-delimited format, and starts with the room id we're describing, followed by the room ids reached when the user goes 'up', 'down', 'north', 'south', 'east', and 'west', respectively. This pseudocode example uses the number 0 to indicate it is impossible to move in the corresponding direction.

You can implement this configuration language however you'd like. For example, this configuration could be placed in a text file and read by the adventure game when it starts. Maybe you'd like to combine the rooms, their descriptions, and their connections into a single, unified configuration. Or perhaps you'd like to just use data structures written in the language you have chosen to write the game in.

Finally, let's say we have chosen a CSV text file that our Python game engine reads. Then we might run the game like:

```
$ python adventure.py hobbit.csv
```

The Game REPL

The game wouldn't be much fun if we couldn't play it! So we would like to provide the user a way to look at the room descriptions and move around between them. For Nano Adventure, the user should be able to use these commands:

```
look:  show the description of the room
north: attempt to move north
south: attempt to move south
east:  attempt to move east
west:  attempt to move west
up:    attempt to move up
down:  attempt to move down
quit:  stop playing!
```

Assuming the user starts in the first room, here's what playing the Nano Adventure might look like:

```
$ python adventure.py hobbit.csv

Welcome to Nano Adventure!

You stand just outside the entrance of a hobbit hole.
> west

You are in the entrance hallway of the hobbit hole. A wizened wizard
and small hobbit stand nearby looking curiously at you.
> east

You stand just outside the entrance of a hobbit hole.
> east

You cannot move east.
> look

You stand just outside the entrance of a hobbit hole.
> quit

Goodbye!
```

Tips

- Keep the code simple. This isn't a test of your expertise in the language, or your ability to implement a general artifical intelligence!
- If there's something you don't understand because the description of the game above isn't clear or complete, make a good guess what you think it should be, and try something. Don't worry that you might "get it wrong".
- Comments explaining *why* you did something that might not be obvious are always helpful.
- Provide a list of any dependencies that need to be installed in order to run your game.
- Provide a commandline to run your unit tests.
- Provide a commandline for starting your game.
- Have fun with it! Funny extra 'features' may earn bonus points. :)