ICS 53, Winter 2020

Assignment 1

You will write a text adventure game. The idea of a text adventure game is that the player is in a virtual room in a "dungeon", and each room has a text description associated with it, such as, "This room in big and brightly lit. It has doors on the north and west walls". The player can move from room to room using compass directional commands.

Your program will be a function called `adventure` which takes no arguments and returns no values. Your function `adventure` will allow the user to enter commands related to moving through the dungeon and interacting with objects. Your function should print out a prompt to indicate that the user can enter a command. Your prompt should be a '$' character. At the prompt the user will type a command followed by a set of arguments for the command. Your function will perform whatever action is indicated by the command, and then your program will print a '$' character on a new line in order to let the user know that he/she can type a new command. If the command is a compass command which moves the player into a new room, your program will print the description of the room before printing the '$' character on the next line.

Dungeon File

The rooms in the dungeon, and their arrangement, is stored in a *dungeon file*. When the game starts, the player will have to issue a command to load a dungeon file describing the dungeon. The dungeon file will be a text file containing information about each room in a specific format. Each non-empty line of the file will contain information about a single room in the dungeon. Each room is associated with the following 6 fields of information.

1. ***Room number***: This is a positive integer which uniquely identifies the room.
2. ***Description***: This is a string which is printed when the player enters the room. Each string is delimited with the '+' character rather than a traditional quote. This means that a '+' character cannot be included inside the string.
3. ***North room***: This is the room number of the room immediately to the north of this room. If there is no room to the north of this room then this value is -1.
4. ***South room***: This is the room number of the room immediately to the south of this room. If there is no room to the south of this room then this value is -1.
5. ***East room***: This is the room number of the room immediately to the east of this room. If there is no room to the east of this room then this value is -1.
6. ***West room***: This is the room number of the room immediately to the west of this room. If there is no room to the west of this room then this value is -1.

All 6 fields of information about each room are contained in order on a single line in the dungeon file. Each field of information is separated by one or more spaces. The room described on the first line of the file is the initial room where the player will start after the dungeon file is loaded.

Here is an example dungeon file. Be aware that this is only an example and that your code must work with any valid dungeon file.

```
1 +This is an open field west of a white house, with a boarded front door.+ 2 -1 -1 -1
2 +North of House: You are facing the north side of a white house.+ -1 1 3 -1
3 +Behind House: You are behind the white house.+ -1 4 -1 2
4 +South of House: You are facing the south side of a white house.+ 3 -1 -1 -1
```

Your program should accept the following command related to dungeon files:

1. **loaddungeon**: This command reads a dungeon file, allowing the player to explore the dungeon. The function takes one argument, the name of the dungeon file. The function returns no values. The loaddungeon command can only be issued once. If the player tries to issue the command a second time, the error "Dungeon already loaded" should be printed.

Moving Through the Dungeon

The program must keep track of the *current room* which is the room which the player is currently in. After executing the loaddungeon command, the *current room* should be the room described on the first non-empty line of the dungeon file. The player will enter compass direction commands to move from one room to another. Your program should accept the following commands related to movement:

1. **north**: This command moves the player into the room to the north of the current room. If there is a room to the north then the description of the new room is printed. If there is no room to the north then the message, "You can't go there." is printed to the screen.
2. **south**: This command moves the player into the room to the south of the current room. If there is a room to the south then the description of the new room is printed. If there is no room to the south then the message, "You can't go there." is printed to the screen.
3. **east**: This command moves the player into the room to the east of the current room. If there is a room to the east then the description of the new room is printed. If there is no room to the east then the message, "You can't go there." is printed to the screen.
4. **west**: This command moves the player into the room to the west of the current room. If there is a room to the west then the description of the new room is printed. If there is no room to the west then the message, "You can't go there." is printed to the screen.

Command Summary

This is a summary of all of the commands that your program should accept.

1. **loaddungeon**: This command reads a dungeon file, allowing the player to explore the dungeon. The function takes one argument, the name of the dungeon file. The function returns no values. The loaddungeon command can only be issued once. If the player tries to issue the command a second time, the error "Dungeon already loaded" should be printed. After executing the loaddungeon command, the *current room* should be the room described on the first non-empty line of the dungeon file.
2. **north**: This command moves the player into the room to the north of the current room. If there is a room to the north then the description of the new room is printed. If there is no room to the north then the message, "You can't go there." is printed to the screen.
3. **south**: This command moves the player into the room to the south of the current room. If there is a room to the south then the description of the new room is printed. If there is no room to the south then the message, "You can't go there." is printed to the screen.
4. **east**: This command moves the player into the room to the east of the current room. If there is a room to the east then the description of the new room is printed. If there is no room to the east then the message, "You can't go there." is printed to the screen.
5. **west**: This command moves the player into the room to the west of the current room. If there is a room to the west then the description of the new room is printed. If there is no room to the west then the message, "You can't go there." is printed to the screen.
6. **quit:** This command should end the program.

Example Output

The following is a running example showing how your program should respond to commands. The text typed by the user is in bold. In the example below, we assume that the contents of `dfile.txt` are the example dungeon file shown above in the **Dungeon File** section of the homework.

```
$ loaddungeon dfile.txt
This is an open field west of a white house, with a boarded front
door.
$ north
North of House: You are facing the north side of a white house.
$ east
Behind House: You are behind the white house.
$ south
South of House: You are facing the south side of a white house.
$ quit
```

Implementation Notes

- Assume that each line in the dungeon file can contain no more than 120 characters.
- Assume that the dungeon file can contain no more than 100 lines.

Submission Requirements

Place all of your code in a single C file which you will upload to Canvas. Be sure that your program compiles on openlab.ics.uci.edu using gcc version 4.8.5.