CS 172 – Lab 7: Linked Lists

Introduction

We have discussed how we can use linked lists to move around in memory (or at least how to store a lot of things by storing them in different chunks of memory which are linked together). Still, it's hard to visualize what is actually happening when we move around a linked list. To help with this, in this lab, we will make a maze. The maze is based on the idea of linked lists. But instead of a node (which for our maze will be rooms) having one possible link/direction (next), we will have 4 possible directions (north, east, south, and west).

The Classes

Milestone 1: The `Room` Class

You are provided a template for a room module with this assignment. This module contains a `Room` class.

A `Room` will be the basic object for our maze (think of it as a `Node`). A `Room` can have 4 doors/links (pertaining to north, south, east, and west). Attached to each these directions we have either another `Room` or `None` (we could also imagine that the doors that lead to `None` are just walls, or locked doors).

We want the player to be able to tell what room they are in. Therefore, each `Room` will have a unique description. When the player enters a room, the program will describe the room. This way the player will know if they went back to a room that have already been to.

You **MAY NOT** change the method's parameters or names in **ANY** way.

The public interface of the `Room` class is as follows:

- Constructor:
  - `def __init__(self, descr)` : where desc is a string that contains the description of the room.
- Getters:
  - `def __str__(self)` : Returns a string description of the room.
  - `def getNorth(self)` : Returns the object north of this room (which should be either another `Room` or `None`)
  - `def getSouth(self)` : Returns the object south of this room (which should be either another `Room` or `None`)

    - `def getEast(self)`: Returns the object east of this room (which should be either another `Room` or `None`)
    - `def getWest(self)`: Returns the object west of this room (which should be either another `Room` or `None`)
- Mutators
    - `def setDescription(self, d)`: Sets the description of the `Room` object to the string `d`.
    - `def setNorth(self, n)`: Sets the north link of the room to the object `n` (which should be either another `Room` or `None`)
    - `def setSouth(self, s)`: Sets the south link of the room to the object `s` (which should be either another `Room` or `None`)
    - `def setEast(self, e)`: Sets the east link of the room to the object `e` (which should be either another `Room` or `None`)
    - `def setWest(self, w)`: Sets the west link of the room to the object `w` (which should be either another `Room` or `None`)

Your task here is to finish the implementation of the provided methods as described above. Test that you are able to instantiate the class and use its methods.

Once you have completed this part, this is a good point to submit your code and earn 15 points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Milestone 2: The Maze Class

A maze is just a bunch of rooms (similar to how a Linked List is just a bunch of nodes, albeit connected). There are three things we need to know about a maze:

1. We need to know the start of the maze. That is, what room the player starts in.
2. We need to know where the exit is. This will allow us to tell the player that they exited the maze.
3. We also need to know what room the player is currently in.

You are also provided a template for the `Maze` class within a maze module. Once again, you **MAY NOT** change the method's parameters of names in **ANY** way.

The `Maze` class's public interface is as follows:

- Constructor:
    - `def __init__(self, st = None, ex = None)`. Sets the starting room (think head of the Linked List) to the `st` object and the

exit room to the `ex` object. Both of these parameters should be either a `Room` object or a `None` object.

- Getters:
  - `def getCurrent(self)`: Returns the current room the player is in.
  - `def atExit(self)`: Returns a `Boolean` telling us if the player is currently in the exit room or not.
- Mutators:
  - `def moveNorth(self)`: Attempts to move the player to the room north of the room they are currently in. If this is a `None` object it tells them that this door is locked (see example text later) and does not move them. If it is a valid `Room` it moves them there and tells them what the room is (prints its description). If this room is the exit, it tells them this as well.
  - `def moveSouth(self)`: Attempts to move the player to the room south of the room they are currently in. If this is a `None` object it tells them that this door is locked (see example text later) and does not move them. If it is a valid `Room` it moves them there and tells them what the room is (prints its description). If this room is the exit, it tells them this as well.
  - `def moveEast(self)`: Attempts to move the player to the room east of the room they are currently in. If this is a `None` object it tells them that this door is locked (see example text later) and does not move them. If it is a valid `Room` it moves them there and tells them what the room is (prints its description). If this room is the exit, it tells them this as well.
  - `def moveWest(self)`: Attempts to move the player to the room west of the room they are currently in. If this is a `None` object it tells them that this door is locked (see example text later) and does not move them. If it is a valid `Room` it moves them there and tells them what the room is (prints its description). If this room is the exit, it tells them this as well.
  - `def reset(self)`: Sets the player's current room to be the start room.

Your task here is to finish the implementation of the provided methods as described above. Test that you are able to instantiate the class and use its methods.

Once you have completed this part, this is a good point to submit your code and earn 15 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

The Main Application

Milestone 3: a basic script
In your **main.py** file add a few lines of code to show that you are able to:

1. Create an empty Maze.
2. Create a few rooms, link them, and create a second maze using these rooms.

Once you have completed this part, this is a good point to submit your code and earn 20 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Milestone 4: The `play` function
In your **main.py** file create a function called `play(Maze)` that takes a `Maze` object as its only parameter. Given a `Maze` object, the function will do the following:

1. Print a description of the room.
2. Ask the player what direction to move, in the format: `Enter direction to move north west east south restart`
3. Either move into the next room or stay in the same place if the path is blocked, telling them `Direction invalid, try again.`
4. If they moved to the exit, print `You found the exit!` and end the script. If they did not find the exit, return to step 1.

Once you have completed this part, this is a good point to submit your code and earn 10 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Create some Mazes
Next, we want to test your ability to make mazes! In your main script create two mazes called **`SIMPLE_MAZE`** and **`INTERMEDIATE_MAZE`** . **The names MUST match exactly**. Our tests will import these mazes that you created and will try to apply a set of movements to solve them.

Below is the architecture of these mazes:

Milestone 5: SIMPLE_MAZE: This maze consists of three rooms. The maze should be solved when the movements *east* and *north* are applied in that order. This means you arrive at the exit when you go east room and then the north room.

The description is just `Room<<X>>` where `<<X>>` is the number of the room (for example: **Room1**).

Once you have completed this part, this is a good point to submit your code and earn 10 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Milestone 6: INTERMEDIATE_MAZE: This maze consists of six rooms. This maze should be solved when the movements are *west*, *west*, *west*, *north*, *east*. This means you arrive at the exit when you go west room, then west room again, then west room again, then take north and then finally the final east room. At the end of the movements, `atExit` should be `True` when it is called. The description is just `Room<<X>>` where `<<X>>` is the number of the room.

Once you have completed this part, this is a good point to submit your code and earn 10 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Here's an example of creating a maze with three rooms such that if the player goes **north then east** they arrive at the exit (Note: this doesn't adhere to the room naming convention mentioned above):

```
#create the rooms
room1 = Room("This room is the entrance.")
room2 = Room("This room has a table. Maybe a dining room?")
room3 = Room("This room is the exit. Good Job.")

#room 2 is north of room 1. Make sure to connect them both ways
#(it's not a 1-way door!)
room1.setNorth(room2)
room2.setSouth(room1)

#room 3 is east of room 2. Make sure to connect them both ways
#(it's not a 1-way door!)
room2.setEast(room3)
room3.setWest(room2)

#make a maze
myMaze = Maze(room1, room3)
```

If you want to play your maze you could then run:

```
play(myMaze)
```

Here is an example play of the above maze.

```
This room is the entrance.
Enter direction to move north west east south restart
west
Direction invalid, try again.
This room is the entrance.
Enter direction to move north west east south restart
north
You went north
This room has a table. Maybe a dinning room?
Enter direction to move north west east south restart
reset
You went back to the start!
This room is the entrance.
Enter direction to move north west east south restart
north
You went north
This room has a table. Maybe a dinning room?
Enter direction to move north west east south restart
west
Direction invalid, try again.
This room has a table. Maybe a dinning room?
Enter direction to move north west east south restart
east
You went east
You found the exit!
```

Milestone 7: Play your INTERMEDIATE_MAZE

Finally, in your main script play your INTERMEDIATE_MAZE via `play(INTERMEDIATE_MAZE)`. We will check your output against expected output for a few moves.

Once you have completed this part, this is a good point to submit your code and earn 20 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Scoring

The score for the assignment is determined as follows:

- 10 points: Script successfully creates an empty maze as Maze()
- 10 points: Script successfully creates a few rooms, links them, then creates a maze using them.
- 10pts: SIMPLE_MAZE: Movements east and north successfully solves this maze.

- 10pts: INTERMEDIATE_MAZE: Movements west, west, west, north, east successfully solves this maze.
- 15pts: Room class is implemented correctly.
- 15pts: Maze class is implemented correctly.
- 10pts: play() function is implemented correctly and works as expected.
- 10pts: Program runs without problems.
- 10 points: Program style: header comment, function docstring, good variable names.

Note: Please make sure you put write both lab partners' names and userIDs (in the form abc123) in the header comment of the file you submit for this assignment.