Lab 8

Goals-
Build a linked structure using pointers

You will build a maze using dynamic memory allocation and allow the user to move through the maze. You will use a struct for a room and pointers to connect the rooms (chambers?). All movement will be in one of 4 directions; east, west, north, or south.

The struct will need 4 pointers, one for each possible direction. A NULL pointer will indicate that there is no doorway/arch in that direction. One room will be the starting location. One other chamber must have a special door that is the exit.

The struct will include a label for the room. When the user/player enters a room it will display the label. When you first start the game you will provide an option to turn the labels off. Labels off makes a more challenging puzzle. Labels on will make grading easier. Remember that even with labels they do not need to be helpful.

The maze is not required to fit on a two dimensional matrix. For example, you can four rooms:

```
        A       B       C       D
```

A>east leads to B, B>east leads to C, C>east leads to D, and one (or all) of D's doors leads to B. Links can allow you to be creative. ☺

You can be even more creative. Make it a puzzle. You can put the exit in the starting chamber, but the player must enter *X* other chambers before they are allowed through that doorway. Or consider using one-way doors. Or the exit is not marked as such and when the user passes through there can be a simple message telling them that they have finished. But do not be TOO creative and put a trap in your maze.

The interface will be simple. When the program starts you give them the option to display the room labels. Then you display the movement prompt:

```
  You are in room 3.14.  You can move East or North.  Which
direction do you want to try?
```

Or more enigmatic:

```
     You are in a room.  Which way do you want to try?
```

If they select a direction with a NULL pointer display an 'error' message such as:

```
     You cannot go that way.
```

When they reach the exit you can display suitable "celebratory" message such as:

```
  You made it out!!  Mysteriously, you are 25 years older
and no one remembers you!
```

It is your choice to include an option to quit the program without reaching the exit.

The basic structure is specified. No pun, but it is the struct for a room. There are still other considerations for your design. Considering testing. With links you want to ensure you test all of them! You can do this one room at a time, go east and return. If you have one-way doors then the sequence of rooms for that test might be longer. For testing, by you and your grader, you MUST include a map of your maze. Otherwise a grader may not be able to determine if a link is not working, or there simply is no door there. It must clearly show where each door is and where it leads.

De-allocate memory before closing your program.

HINT: As always use incremental development! This may be more important with dynamic structures and/or links. Start with the entrance and one door that is the exit. Once that is working add doors and test that they work both ways. Then you can probably start adding more than one room at a time. But remember, the runtime errors can involve segmentation faults or core dumps, which provide no information about where the problem occurs. Save the puzzle for last. So move slowly. ☺

Modular Grading

We are using modular grading. Each lab will be divided into specific modules. Each module will be graded pass/fail. It either works properly or it does not. 10% of every lab or assignment grade is style/comments or other elements of self-documenting code and clarity. Remember the labs are worth 10 points total.

Programming style- 1 point

Create the room struct that uses links properly- 2 points

Create a maze of at least 4 rooms- 3 points

Correct user input/output - 2 points

Proper use of dynamic memory- 1 point

Brief description of how you tested the structure- 1 points