COMPUTER SCIENCES AND SOFTWARE ENGINEERING
AUBURN UNVERSITY

COMP 2710
Software Construction
Section 002

Fall 2017

Lab 3
Multi-Player
Snakes and Ladders Maze Game

Due: November 15, 2017

Points Possible: 100
Due: Written Portion: November 6th, 2017 by 11:55 pm. Submission via Canvas (25 points)
Program: November 15th, 2017 by 11:55 pm via Canvas (75 points)

No late assignments will be accepted.
No collaboration between students. Students must NOT share any project code with each other. Collaborations in any form will be treated as a serious violation of the Auburn University's academic integrity code.

Goals:

- To develop an application for multiple players to find paths through a maze
- To develop a maze application that use graph abstraction
- To learn the use of pointers and dynamic memory allocations
- To learn about how to manipulate graph data structures
- To perform Object-Oriented Analysis, Design, and Testing

Software Process (Steps 1-3 due with initial turn-in 25 points, steps 4-5 due with final turn-in 75 points):
Create a text, doc, or .pdf file named "<username>-3p" (for example, mine might read "lim-3p.txt") and provide each of the following. Please submit a text file, a .doc file or .pdf file (if you want to use diagrams or pictures, use .doc or .pdf). You are free to use tools like Visio to help you, but the final output needs to be .txt, .doc, or .pdf.

1. **Analysis:** Prepare use cases. Remember, these use cases describe how the user interacts with a maze system (what they do, what the systems does in response, etc.). Your use cases should have enough basic details such that someone unfamiliar with the system can have an understanding of what is happening in the maze system interface. They should not include internal technical details that the user is not (and should not) be aware of.
2. **Design**:
   a. Create a Class Diagram (as in Lab 2). Be sure to include:
      1) The name and purpose of the classes
      2) The member variables and the functions of the class
      3) Show the interactions between classes (for example, ownership or dependency)
      4) Any relevant notes that don't fit into the previous categories can be added
   b. Create the data flow diagrams. Show all the entities and processes that comprise the overall system and show how information flows from and to each process.

3. **Testing**: Develop lists of *specific* test cases OR a driver will substitute for this phase:
      1) For the system at large. In other words, describe inputs for "nominal" usage. You may need several scenarios. In addition, suggest scenarios for abnormal usage and show what your program should do (for example, entering a negative number for a menu might ask the user to try again).
      2) For each object. (Later, these tests can be automated easily using a simple driver function in the object)

4. Implement the Snakes and Ladders Maze game system

5. **Test Results**: ***After developing the Snakes and Ladders Maze Game system***, actually try all of your test cases (both system and unit testing). Show the results of your testing (a copy and paste from your program output is fine – don't stress too much about formatting as long as the results are clear). You should have test results for every test case you described. If your system doesn't behave as intended, you should note this. Note: Driver output will substitute for this phase.

Program Portion:

The graph abstraction is an important one since it is used in many different areas of computer sciences and software engineering. For instance, the Internet makes use of graphs representing the Internet router network configuration in order to determine the best route for forwarding packets from the source at one end of the Internet to the destination at the other end. Once a graph is constructed to represent the Internet configuration accurately, then the router executes an algorithm to determine the shortest path from each node to each destination and stores the information in its routing table.

In Lab 3, you will have fun constructing a similar graph to represent a **pointer-based maze**, with snakes and ladders, which can be used to implement a simple *multi-player game*. You will analyze, design, implement and test the pointer-based maze game system. Once the graph is constructed by the game system, *multiple players* will try to traverse the maze from a starting node to a finish node, by selecting the direction to take at each node. Each player will take turn in traversing the graph. During a player's turn, he/she will roll a dice that will determine the number of nodes that he/she visits during that turn. ***As the player traverse the graph, your program must follow the pointer to each node.*** Since the maze configuration is hidden from the player, he/she will first randomly select each direction and learn and remember past dead-end routes and try to find the correct route to the finish point. When the player reaches the finish point, the program will print out the number of steps taken by the player and the *all the nodes visited by the player in the correct sequence,* starting from the start node to the destination node.

Multiple Players and Random Dice throw?

Your system must allow at least two players to play in this game and compete to be the first player to traverse the maze from a starting node to a finish node, by selecting the direction to take at each node. Initially, the system will prompt the users to enter the number of players and the names of each player. Then each player will take turn in traversing the graph, starting from the first player.

Each player will take turn in traversing the graph as follows. During a player's turn, he/she will roll a dice that will determine the number of nodes that he/she visits during that turn. When he/she is at a node, the system will list all the legitimate directions that he/she can follow, from which he/she will select one direction to traverse. Depending on the dice throw number, the player must visit that number of nodes. An exception is at the end. A player wins whenever he/she is the first player to reache the destination node even before the number of nodes that they are supposed to visit.

To simulate the random dice throw, you must use the random number generator function `rand()`. However, you must use the correct expression involving `rand()` that will return a value from 1 to 6. In order to ensure that the function `rand()` returns a truly random number each time you execute your program, you must set a different seed for the random number generator using the function call `srand(time(0))`, where `time(0)` is a function call that returns the current time in terms of number of seconds since January 1, 1970.

What is a graph?

A graph is consists of multiple nodes and edges. For instance, a graph can be represented by Figure 1 below, where the circles represent the nodes and the lines represent the edges. An edge connects two nodes.

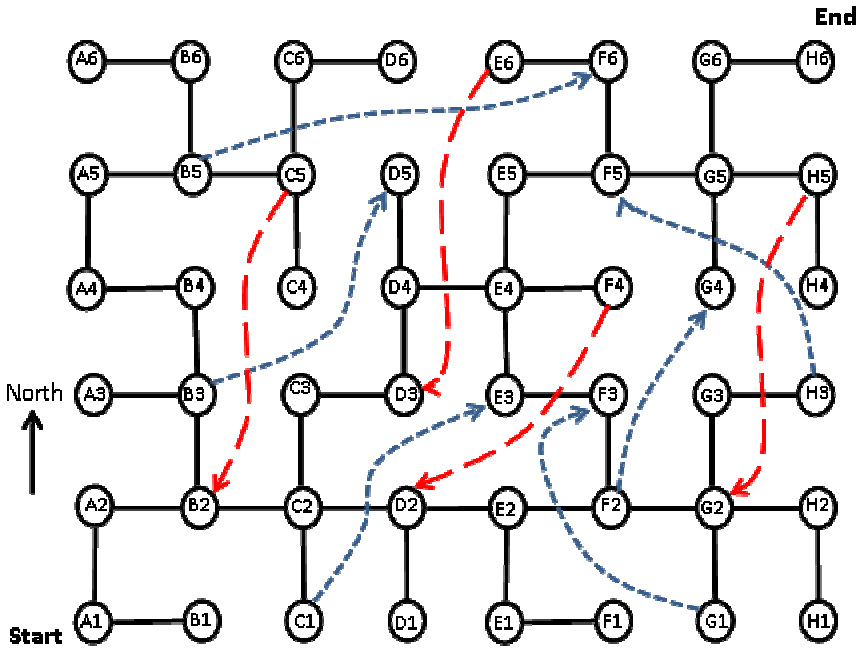

Figure 1. A Sample Maze

In this lab assignment, we will only consider special graphs, where each node can have four normal edges, in the four directions: North, East, South and West. ***In addition, each node may also have at most one additional edge for either a snake or ladder that will link that node to another node that is not in the vicinity.*** An implementation detail to note here is that although the edge is represented by a single line, there are actually two links in both directions between the two nodes. For example, the line between A1 and B1 in Figure 1 is implemented by a link from A1 to B1 and another link from B1 to A1. *The snake and ladder edges are only unidirectional*.

What is the Snakes and Ladders Maze Game system?

In the Snakes and Ladders Maze game, there is a maze of rooms and passages connecting the rooms. The maze of rooms and passages can be represented by a graph, such as the graph shown above. The nodes can be thought of as rooms and an edge represents a passage that connects one room to another. Each room can be connected normally to four rooms in the four directions: North, East, South and West, *but may also be connected through secret passage ways to distant rooms*, as represented by the graph above.

In this lab assignment, each player will take turn in traversing the graph. During a player's turn, he/she will roll a dice that will determine the number of nodes that he/she visits during that turn. For traversing the maze, you will implement the maze above using references to instances of a `Node` class, which you will define. Each node in the graph will correspond to an instance of the `Node` class. The edges correspond to the links that connect one node to another and can be represented in `Node` as a pointer variable that points to another `Node` class object. Since there are four possible normal links, `Node` must contain four links (pointers) to other `Node` objects in the normal directions, but may have an additional link (pointer) to a distant node. Your `Node` class must be defined as follows (additional features may also be defined):

```
class Node {
public:
        Node(string newname);
        Node();
        void setNodeName(string newname);
        string getNodeName();
        void attachNewNode(Node *newNode, int direction);
        Node *getAttachedNode(int direction);
        void attachSnakeLadderNode(Node *newNode);
```

```
        Node *getSnakeLadderNode();

private:
        string name;
        Node *attachedNodes[4];
        Node *snakeOrLadderNodes;
};
```

Your graph will consist of `Node` objects that are linked together through pointers. The member variable `attachedNodes` list will contain pointers to nodes in the four directions, for e.g. `attachedNodes[0]` will contain a pointer to the node in the North direction, `attachedNodes[1]` will contain a pointer to the node in the East direction and so forth. The member variable `name` will contain the name of the node, e.g. 'A1'. You must implement the constructors `Node(char newname)`, `Node()` and all the functions: `setNodeName(char newname)`, `getNodeName()`, `attachNewNode(Node *newNode, int direction)`, `getAttachedNode(int direction)`, `attachSnakeLadderNode(Node *newNode)`, and `getSnakeLadderNode()`. In the beginning, ***you must create all the nodes using a dynamic array of `Node` objects***, where the size of the dynamic array is the number of nodes in the graph as provided in the configuration file.

The `Start` variable is a pointer that points to the node where the player starts, which may represent Node `A1`. The goal is to reach the destination which is the node that is referenced by the `Destination` variable, for e.g. it may point to the Node `H6`.

Graph Configuration

The configuration of the graph that you will use in the program will be read in from a text file called the configuration file. The first line of the file specifies the number of nodes in the graph. The next line specifies the name of the starting node and the following line specifies the name of the destination node. Starting from the fourth line, each line in the rest of the file indicates information on each node, i.e. the name of the node, the links that may exist from that node to another node in the North, East, South and West directions and another possible link for a snake or ladder. If there is no link in a specific direction, then there is an '*' in place of the node name. For example, the first eleven lines of the configuration file for the graph in Figure 1 is as follows:

```
48
A1
H6
A1  A2  B1  *  *  *
B1  *  *  *  A1  *
C1  C2  *  *  *E3
D1  D2  *  *  *  *
E1  E2  F1 *  *  *
F1  *  *  *  E1  *
G1  G2  *  *  *  *
H1  H2  *  *  *  *
…….
```

In the fourth line of the configuration file above, the name of the node is A1. It has a pointer to node A2 in the North direction and a pointer to node B1 in the East direction but no link in either the South or West direction and no snake or ladder link. Thus the South, West, and Snake/Ladder pointers are set to NULL.

Your program will first read the graph configuration file and construct the graph data structures used for the maze game. Your program must run correctly with any graph configuration file. Several test configuration files will be released to you close to the deadline for you to test the correct execution of your program. *A graph configuration file will have any number of nodes.*

In the Snakes and Ladders Maze Game, each player will traverse the graph based on the inputs given by the players. The number of nodes that a player visits during his/her turn depends on the number on the dice that he/she rolled. When the player is at a current node, the program will display the allowable moves in the North, East, South or West directions. The player will then enter the desired direction. Your program must traverse to the next node by following the `attachedNode` pointer using the `getAttachedNode()` function. *If a node that a player entered has a Snake or Ladder link (pointer), i.e. the snakeOrLadderNode pointer is not NULL, then the player will automatically and immediately be placed at the node that is*

pointed to by the snakeOrLadderNode pointer. While the program traverses the graph, it also counts the number of steps and keeps track of the nodes that it visited. Upon reaching the finish point, it will print out the congratulation banner, the number of steps taken and the sequence of nodes that the player traversed from the Start node to the Destination node.

The user interface must check for correct input value from the users. If there is any error, e.g. selecting an invalid direction, then the program must display the appropriate error message and continue to prompt for the correct input. Your program must not exit or terminate when there is an incorrect input value.

The name of your program must be called <username>_3.cpp (for example, mine would read "lim_3.cpp"

Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. You will lose points if you do not use the specific program file name, or do not have a comment block on **EVERY** program you hand in.

Your program's output need not exactly match the style of the sample output (see the end of this file for one example of sample output).

Important Notes:
You must use an object-oriented programming strategy in order to design and implement this pointer-based maze game system (in other words, you will need to write class definitions and use those classes, you can't just throw everything in main()). A well-done implementation will produce a number of robust classes, many of which may be useful for future programs in this course and beyond. Some of the classes in your previous lab project may also be re-used here, possibly with some modifications.
Remember good design practices discussed in class:

a) A class should do one thing, and do it well
b) Classes should NOT be highly coupled
c) Classes should be highly cohesive

- You should follow standard commenting guidelines.

- You DO NOT need any graphical user interface for this simple, text-based application. If you want to implement a visualization of some sort, then that is extra credit.

Error-Checking:

You should provide enough error-checking that a moderately informed user will not crash your program. This should be discovered through your unit-testing. Your prompts should still inform the user of what is expected of them, even if you have error-checking in place.

Please submit your program through the Canvas system online. If for some disastrous reason Canvas goes down, instead e-mail your submission to TA – Yue Cui – at yzc0058@tigermail.auburn.edu. Canvas going down is not an excuse for turning in your work late.

You should submit the two files in digital format. No hardcopy is required for the final submission:

<username>_3.cpp
<username>_3p.txt (script of sample normal execution and a script of the results of testing)

A sample execution is shown below, where the bold fonts indicate input by the user.

> lim_3

```
        ======================================================
        |   Welcome to the Snakes and Ladders Maze Game      |
        ======================================================

Enter the number of players: 2
Enter the name of Player #1: Jane
Enter the name of Player #2: John

        =====================
        |   Jane's turn     |
        =====================

Jane's turn to throw the dice, just hit enter
Jane's Dice throw is 4
Jane can move to 4 nodes

Jane is currently in Node A1 of the Maze Game, you can go North or East. What is your choice? N

Jane is currently in Node A2 of the Maze Game, you can go East or South. What is your choice? E

Jane is currently in Node B2 of the Maze Game, you can go North or East or West. What is your
choice? E

Jane is currently in Node C2 of the Maze Game, you can go North or East or South or West. What
is your choice? S

        =====================
        |   John's turn     |
        =====================

John's turn to throw the dice, just hit enter
John's Dice throw is 3
John can move to 3 nodes

John is currently in Node A1 of the Maze Game, you can go North or East. What is your choice? N

John is currently in Node A2 of the Maze Game, you can go East or South. What is your choice? E

John is currently in Node B2 of the Maze Game, you can go North or East or West. What is your
choice? N

        =====================
        |   Jane's turn     |
        =====================

Jane's turn to throw the dice, just hit enter
Jane's Dice throw is 2
Jane can move to 2 nodes

Jane is currently in Node C1 of the Maze Game, and you have taken a Ladder to Node E3, you can
go North or East. What is your choice? N

Jane is currently in Node E4 of the Maze Game, you can go North or East or South or West. What
is your choice? N

        =====================
        |   John's turn     |
        =====================

John's turn to throw the dice, just hit enter
John's Dice throw is 1
```

John can move to 1 node

John is currently in Node B3 of the Maze Game, and you have taken a Ladder to Node D5, you can go South only. What is your choice? **S**

```
        ======================
        |   Jane's turn      |
        ======================
```

Jane's turn to throw the dice, just hit enter
Jane's Dice throw is 5
Jane can move to 5 nodes

Jane is currently in Node E5 of the Maze Game, you can go East or South. What is your choice? **E**

Jane is currently in Node F5 of the Maze Game, you can go North or East or West. What is your choice? **E**

Jane is currently in Node G5 of the Maze Game, you can go North or East or South or West. What is your choice? **N**

Jane is currently in Node G6 of the Maze Game, you can go East or South. What is your choice? **E**

Congratulations, Jane is the winner! You have reached the destination point.
You took 11 steps.
The nodes you visited are: A1 A2 B2 C2 C1 E3 E4 E5 F5 G5 G6 H7