CSCI 135 Section 2 - Fall 2014
Programming Project 1

Due: Wednesday, October 29th, 2014, 11:59 pm

1 Introduction

This project will introduce you to reading files and working with arrays in C++. We discussed file reading in class, but it is recommended that you re-read the textbook's section in Chapter 2 on using the `ifstream` object to read files. Some explanation and sample code follows. This sample shows how to read a file of integers stored as text.

```
ifstream infile;
infile.open("numbers.txt");
int num, x = 0;
while(infile >> num)
{
    cout << num << " ";
    x++;
}
infile.close();
```

This code snippet opens a file "numbers.txt" for file input/output. A variable `num` is declared, and it will hold the latest number to be read from the file - the statement `infile >> num` does this. However, `>>` can be used to determine whether anything was read from the file. While this does not return a boolean, a mechanism in its implementation allows us to use this in place of a boolean. Therefore, this loop will execute until all integers are read. What do you think will be the value of `x` at the end of this loop?

If we know the number of tokens in the file, this code is equivalent to

```
ifstream infile;
infile.open("numbers.txt");
int num, x;
for(x=0;x<NUM_OF_TOKENS;x++)
{
    infile >> num;
    cout << num << " ";
}
infile.close();
```

1.1 Programming Rules

Under "Course Information" on Blackboard (for CSCI 135), you will see a document called "Programming Rules". Please read it and make sure you follow the rules described therein.

1.2 Commenting

Note that you will lose points if you do not comment your code. Apart from explaining what a block of code is doing, you should also write a brief description of your program at the top of your `.cpp` files using multiline comments (`/* ... */`). Meaningful comments will also assist the instructor in understanding your code easily.

1.3 Design

It's a bad idea to jump right away to coding. Always take a few minutes to think about what the problem is asking for, then start formulating your solution on paper.

Moreover, since the task deals with reading from a file, it would be a good idea to first write code that reads the files. Then incrementally add more functionality.

2 Snakes and Ladders

Snakes and Ladders is an ancient Indian board game, still popular today. In this project you will simulate this game between a number of players. The game is played on an $n \times n$ grid, where the bottom left corner is numbered 1, and the top left corner is numbered n^2. There are 'snakes' and 'ladders' placed across the board, with both ends of a snake or ladder placed in distinct cells. A user's objective is to reach last cell before the other users by rolling a d sided die. If the user lands on a snake's head, they are teleported to the cell containing the snake's tail. If they land on the bottom end of the ladder, they move to the top. For more information, see `http://en.wikipedia.org/wiki/Snakes_and_Ladders`.

Your program will deal with the following condition:

1. The number of players can be between 2 and 5.
2. The size of the square board can be between 8×8 and 12×12.
3. The die can have between 2 and 12 sides.
4. Arbitrarily placed snakes and ladders.
5. If a user rolls a d on a $d-$sided die, he/she gets another to play again.
6. The game ends as soon as one player reaches or crosses the last cell.

Your task is to simulate such a game with the specified number of players in the specified environment. Each player rolls a die, then moves a number of cells. If they land on a snake or ladder, they should move to the target cell. The game should go on until someone lands on or beyond the last cell (i.e., if, on a 100 cell board, the player is

currently on 97 and rolls a 4, they win).

Your program should output the following:

1. Moves played by each player in each round. If they don't land on a snake or ladder, just state "Player [X] rolled a [Y] and moved to [destination cell]". If they land on snake or ladder, state the new destination in addition to this and whether it was due to a snake or a ladder.

2. At the end, it should list the players, sorted by the cells they are currently on, in descending order.

Input

All of this information will be read from a file, called `input.txt`. The first line of the file contains the value of n (so that you have an $n \times n$ board. The second line of the file contains the number of players. The third line contains the number of sides of the die. All subsequent lines have 3 tokens each and describe the positions of the snakes and ladders. Their format is the following: a character 'S' (for snakes) or 'L' (for ladders), followed by a cell number, followed by an "offset". In case of a snake, the cell number corresponds to the snake's mouth, and in case of a ladder, it corresponds to the bottom of the ladder. The offset determines how many cells the player is promoted (in case of a ladder) or demoted (in case of a snake). Note that the number of snakes and ladders is not specified separately. Here is a sample input:

```
10
4
6
S 15 6
L 20 10
L 12 8
S 54 12
L 64 9
S 73 30
L 80 10
S 99 15
```

This file represents a 10×10 board, played by 4 players using a 6-sided die. An `input.txt` is placed on Blackboard as well. However, do test your program with different inputs!

2.1 Approach

These tasks are not difficult, provided you carefully design your program. Please note that although this game is played on a square grid, there is no need to use 2-D arrays in your simulation. You only need to keep track of each player's position. Use an array for this. Use an array to store the snakes and ladders, as well. Make sure you do not create an array that is too big. How many snakes or ladders can the largest board contain? Your array should not be bigger than this.

3 Submission

The format of the filename should be `<YOUR_LAST_NAME>_hw1.cpp`. Replace `<YOUR_LAST_NAME>` with your last name.

Now go to CSCI 135 on Blackboard, go to the "Programming Projects" section, and click on "Project 1". You will see an option to upload your files. Upload both files and click "Submit". If you have something to tell the instructor, enter it in "Comments".