**Data Structures (2028C)**

***Topics covered: Stacks and Queues***

**Objective:**

The objective of this Lab is to examine stacks and queues built using C++. This will take the famous “Towers of Hanoi” game and modify the rules to be a challenge between 2 players.

**Task 1:** Create a stack class that will be used as the basis for the remainder of the lab.

1. Create a new project. You can name this whatever you like.
2. Design a stack class using an array. This class should be a template.
   1. The constructor should include a parameter to indicate the size of the array.
   2. The array holding the data should be an array of pointers.
   3. The push function should accept a pointer and add that pointer to the top of the stack. It doesn’t need to create any memory.
   4. The pop function should return a pointer. It doesn’t need to delete any memory.
   5. The top function should return a pointer to the item on the top of the stack without removing it.
   6. The length function will return an int indicating the number of items in the stack.
   7. The empty function should empty the stack of all contents. It needs to call delete to avoid memory leaks. It doesn’t need to return any value.
   8. The stack should declare a friend function that can access the underlying array for the purposes of printing it to the screen.
   9. Create the implementation code for the above functions as required. **Do not** use cout in the class. Any errors such as overflow, or underflow should throw a custom class error.

Complete this before moving on to task 2.

**Task 2:** Create a single person version of the “Towers of Hanoi” game.

1. Read the introduction to the game at <https://en.wikipedia.org/wiki/Tower_of_Hanoi>.
2. Create a main program that uses stacks to allows the user to interactively play the game.
   1. It needs to determine the number of disks to use in the game.
   2. It needs to provide a way for the user to indicate the tower to move from and to.
   3. It needs to prevent illegal moves as defined in the Wikipedia rules.
   4. It needs to provide the user with a way to see their progress on the screen.
3. Test your program to ensure it is working correctly. This should test both expected success conditions and expected error conditions.
4. Include in the lab report a screen shot(s) of the output of your tests.

**Task 3:** Make a game of it.

1. Create a new project.
2. Copy any code from the previous project you may need into this project.
3. In this version, there are 2 towers on each side and 1 tower in the middle. Each player needs to move their disks over to the other side (which tower on the other side is not relevant so long as they are on a single tower). You will need to find a way to signify which disk belongs to which player. Each players disk of similar size is considered equal and may be placed on the other. Each player may move any top disk to any tower on their turn so long as the move is legal.
4. Test this game in your group.
5. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of the different strategies you tried (cooperative, defensive, ignoring the other player completely, etc…).

**Task 4:** Create a queue to track steps.

1. Create a new project.
2. Create a queue class. This class should be a template.
3. Design the queue class using a vector to store data. Include the standard functions for a queue.
4. Implement the queue class.
5. Design a data structure to hold a single move.
6. Create an instance of the queue class to hold the move data structure designed above.
7. Copy and modify the code from Tasks 1 and 2 to store any moves made in the queue class. When the game is won, the queue should be used to print out all the moves made in order.
8. Review and modify the code found at <http://proprogramming.org/solution-for-tower-of-hanoi-c/> or any other source you desire to automate the solution for your game.
9. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of how you designed your move data structure and how you had to modify the code for 2

**Lab Submission:**

1. Write a lab report including the following information:
   1. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.
   2. The sections from each task indicated to be included in the lab report.
2. Include all source code from all tasks, input and output files (if any), and any special instructions to compile and run those programs.

**Lab Grading:**

1. 2.0 Points - Lab attendance
2. 1.5 Points - Task 1 has been correctly implemented and meets all requirements.
3. 1.0 Points - Task 2 has been correctly implemented and meets all requirements.
4. 1.5 Points - Task 3 has been correctly implemented and meets all requirements.
5. 2.0 Points - Task 4 has been correctly implemented and meets all requirements.
6. 2.0 Points - Lab report contains all required information and is well written.

If program fails to compile, 0% will be given for that Task.