**Data Structures (2028C) -- Spring 2020 – Lab 7**

***Topics covered: Pointers and Ordered Lists***

*Lab due:* ***Sunday, Mar 8 at 11:55PM for Monday Section***

***Tuesday, Mar 10 at 11:55 PM for Wednesday Section***

**Objective:**

The objective of this Lab is to manipulate an ordered list and explore possible efficiency options in creating the list.

**Task 1:** Create an ordered list class.

1. Create a new project. You can name this whatever you like.
2. Design and implement an ordered list class using an array of pointers as described in class. This class should be a template. The template is expected to have overloaded the >, < and == operators.
   1. The class should have an array that holds 25 items.
   2. The AddItem method should start at the front of the array (index 0) when searching for a place to insert a new item.
   3. The RemoveItem method should ensure that the items in the array are still in order and there are no empty spots between items.
   4. The class should include IsEmpty, IsFull and MakeEmpty methods.
3. Any error conditions encountered such as the item to be removed doesn’t exist or the list is full when trying to add an item should throw a custom error class.

Complete this before moving on to task 2.

**Task 2:** Create a derived class from the class created in Task 1. This version of the class is to investigate the theory that by starting from the middle (index 12), fewer items will need to be moved on every insert

1. This version should have the AddItem method start from the middle of the array when searching for a place to insert a new item.
2. This version may move left or right when inserting but can’t go past the end of the array. If there is still room in the array but not in the direction needed, you will need to shift everything over when inserting.

Complete this before moving on to task 3.

**Task 3:** Create another derived class from the class created in Task 1. This version of the class is to investigate leaving blank spots (points to a null) in the array can reduce the number of moves when inserting.

1. Modify the AddItem method to insert an item halfway between any two items in the array where it belongs. It should only move items in the array if the inserting item sits between two items that are in contiguous locations.
2. Modify the RemoveItem method so that it will not move any items in the array.
3. Include in your lab submission a paragraph for each of the 3 versions of the class a description of what is trying to be achieved, what the strengths and weaknesses are and how you think it will perform.

Complete this before moving on to task 4.

**Task 4:** Create a test program.

1. Create a test program to test the 3 classes performance.
   1. This should instantiate each of the three classes.
   2. This should insert 30 randomly generated items into the array (using integers as the list data type might be the easiest way to do this).
   3. This should remove 25 items from the array.
   4. Individual inserts and removes from steps from b and c should be randomly selected rather than done sequentially.
   5. Every comparison and move operation in the AddItem and RemoveItem methods should be counted (except checks to see if an array location holds a nullptr). The total at the end of the run should be displayed to measure efficiency of the class.
   6. Every insert/remove operation should be identically performed against all 3 of the class instances.
2. Run the program 100 times against the 3 versions of the class. Describe your results and how they compare with your expected results from the end of Task 3
3. Increase the size of the array to hold 50 items. And rerun the program 100 times. Describe your results and how they compare with the previous step.
4. Decrease the size of the array to hold 10 items and rerun the program 100 times. Describe your results and how they compare with the previous step.
5. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of how you designed your program to simplify running it and reporting results 100 times. Discuss your methods for analyzing the results and why you believe those methods are valid. Discuss other methods you discussed as a group and decided not to use.

**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.
3. Package all files in a single zip folder and submit the file to Canvas.

**Lab Grading:**

1. 20% - Lab attendance
2. 10% - Task 1 has been correctly implemented and meets all requirements.
3. 5% - Task 2 has been correctly implemented and meets all requirements.
4. 10% - Task 3 has been correctly implemented and meets all requirements.
5. 35% - Task 4 has been correctly implemented and meets all requirements.
6. 20% - Lab report contains all required information and is well written.

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