**Data Structures (2028C) -- Spring 2017 – Lab 10**

***Topics covered: Hashing and Binary Trees***

*Lab due:* ***Sunday, Apr 1 at 11:55PM for Monday Section***

***Tuesday, Apr 3 at 11:55 PM for Wednesday Section***

**Objective:**

The objective of this Lab is to explore creating and using Hash Tables and Binary Trees.

**Task 1:** Create a Hash Table that uses linear probing.

1. Create a new project. You can name this whatever you like.
2. Design and implement Hash Table class using linear probing as described in class. This class should be a template. The template is expected to have overloaded the == operator and the string object conversion[[1]](#endnote-1) (you don’t need to write these overloads as part of your Hash table, it will be in the data type you are storing in your hash table).
   1. The class should have the following methods fully implemented.
      1. Constructor – This should have an overload indicating the maximum number of items the Hash table can store. The default value is 100.
      2. Hash – This should be a private function that accepts a string and returns an int. For this, we are going to add the sum of all the ASCII values for the passed in string and modulus by the maximum size of the table.
      3. AddItem – adds an item from the list.
      4. RemoveItem – searches the list for the given item. If found, it removes it from the list and returns it. If not found, it returns a null pointer.
      5. GetItem – searches the list for a given item. If found, it returns a pointer to the item but doesn’t remove it from the list.
      6. GetLength – returns an int indicating the number of items in the table.
      7. Destructor
   2. All items passed to or from the class should be done so via a pointer rather than by value.
   3. Make sure you don’t have any memory leaks.

Complete this before moving on to task 2.

**Task 2:** Reuse the class from Lab 8, Task 2. You will need to add the overload of the string object conversion. This object conversion should return the MNumber.

Complete this before moving on to task 3.

**Task 3:** Modify your test program from Lab 8, Task 3 to test your Hash Table class. Include screen shots of the outputs of at least 2 tests in your lab report. Complete this before moving on to task 4.

**Task 4:** Create a derived class from your Hash table that implements a chained hash table. Use your linked list class from Lab 8, Task 1. Modify your test program from Task 3 to test this new class. Include screen shots of the outputs of at least 2 tests in your lab report.

Complete this before moving on to task 5.

**Task 5:** Measure the performance of the linear probing and chained linking implementations of a hash table (similar to Lab 7 task 4). To do this, modify both classes to keep track of the number of times an item is compared in the chain for chained linking[[2]](#endnote-2) and the number of times an additional index is checked for linear probing (you only need to do this for the GetItem). Create a test program that adds 50 randomly generated students (follow appropriate MNumber rules) to both hash tables then calls GetItem on the same student. Record the tracked numbers for both the Hash Tables while performing the above operations. Repeat this measurement with an array size of 150, 200 and 250. Include a table of the results from the 4 trials for the 2 different classes in your lab report.

**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 any special instructions to compile and run those programs.
3. Include a discussion of what modifications you needed to make for Task 2, Task 3 and Task 4.
4. Include a discussion of why you think the results for Task 5 were what they were. This should suggest ideas for further investigation.
5. Submissions must include what each group member has contributed.
6. Package all files in a single zip folder and upload the file to canopy website within the specified due date.

**Lab Grading:**

1. 10% - Lab attendance
2. 10% - Task 1 has been correctly implemented and meets all requirements.
3. 10% - Task 2 has been correctly implemented and meets all requirements.
4. 10% - Task 3 has been correctly implemented and meets all requirements.
5. 15% - Task 4 has been correctly implemented and meets all requirements.
6. 20% - Task 5 has been correctly implemented and meets all requirements. The discussion makes sense and includes suggestions for further investigation.
7. 25% - Lab report contains all required information and is well written.

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

1. See section 14.6 of Gaddis on Object Conversions. [↑](#endnote-ref-1)
2. You may need to modify the Linked List class to achieve this. [↑](#endnote-ref-2)