EECS 560 Lab 5: Hash Table with Open Chaining

Due date:
10/04/2020, 11:59 pm -Sunday

Objective
Get familiar with Hash table with open chaining implementation with C++.Work on basic operations on hash table with open chaining.

Specification of the ADT

1. Refer to section 5.3 in Textbook. Implement the code for methods in figure 5.6 which has Type declaration for separate chaining hash table. The implementation for these methods available in Figure 5.9 and Figure 5.10. Implement the rehash function from figure 5.22 (The part for Rehashing for separate chaining hash table).

2. **For hash function** : Implement myHash function from figure 5.7. Implement hash class required for this myHash function from page 198 , above figure 5.7. Update the template type for hash class as "hashitem".

Additional requirement:

3. Rename the hash class name into "**myHashTable",** instead of "HashTable " as indicated in the textbook.

4. Implement **explicit myHashTable(int size = 101)** defined from textbook template. It was not implemented in textbook.

5. Implement **insert** method of interface **bool insert(HashedObj && x).** This is defined in textbook template but not implemented.

6. Include **myDlList.hpp** in this file to use list operations. myDlList.hpp should be the file implemented in Lab2. Update to use "**myDlList**" instead of "list" in myHashTable class.

7. Implement a **find** method in **myDlList.hpp** to use in hash table (myHashTable class). This function uses MyDlList iterators in parameters and return type.This should have an interface of **typename myDlList<Object> ::iterator find(typename myDlList<Object> ::iterator from, typename myDlList<Object> ::iterator to, const Object & x)**. Update methods in hash table (myHashTable.hpp) to use find method from myDlList.hpp.

8. Modify the **contains** method in interface to **bool contains(const HashedObj & x)** in myHashTable class. To use "find" method from myDlList.hpp.

9. Implement the "**nextPrime**" used in rehash. While rehashing the hash table to a new size, the new size should be the next available prime number after double the size current hash table size. This should have an interface of **int nextPrime(int n).** Where n is integer that is double the size of current hash table size.

10. Implement "**getiteratorbegin**" method, Which returns the begin iterator associated with list at a given hash position. This should have an interface **typename myDlList<HashedObj> ::iterator getiteratorbegin(int index)**. Index is the hash position.

11. Implement "**getiteratorend**" method, Which returns the end iterator associated with list at a given hash position. This should have an interface **typename myDlList<HashedObj> ::iterator getiteratorend(int index)**. Index is the hash position.

12. Implement "**hashsize**" method, Which return the size of current hash table. This should have an interface of **int hashsize().**

Testing and Grading

We will test your implementation using a tester main function, on a number of instances that are randomly generated. We will release the tester main function, several instances (will be different from the grading instances but will have the same format), and expected output together with the lab instruction via Blackboard. You code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.

The command line we are going to use for compiling your code is:

"g++ -std=c++11 main.cpp" (note that main.cpp will try to include the .hpp file you submit, and your .hpp file needs to be property implemented to compile successfully).

Your final score will be the percentage your program passes the grading instances. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline

Please submit your implementation as two different .hpp files, with file names "myDlList_[YourKUID].hpp" and "myHashTable_[YourKUID].hpp". For example, if my KU ID is c123z456, my submission will be files named "myDlList_c124z456.hpp" and "myHashTable_c124z456.hpp". Submissions that do not comply with the naming specification will not be graded. Please submit through Blackboard.