CS 310 Programming Assignment #2

Searching and Hashing

Due: Friday, March 12 12:00 PM (noon)

Overview

For this assignment, you have to design a hash table that is used as the basis for implementing an "in-memory" database of student records.

You have to design and implement methods for

1. Inserting new records into the database
2. Deleting an existing record
3. Printing a record in response to a query

Each record contains the following information

1. Student name (the name has three fields: first name, middle initial, and last name)
2. Student id (a 9 digit string, e.g. "0102239441")
3. Student status (e.g., "Freshman", "Graduate", "Senior", etc.)

A user of the database should be able to type in the student's last name OR the student's ID to retrieve all the information in a student record. Most of the queries on the database use the student's ID to search for the student's information. Thus, the student's ID should be the ***primary key*** for your hash table. However, some of the queries submitted to the system require a search for the student's last name. What this means is that you have to be able to search for a student's record if supplied either the student's last name or the student's id. Your data structures and algorithms should be designed to try and minimize the number of comparisons needed for a search.

An interface for the hash table will be provided to you (in file hash.H); your task is to complete the implementation. **You may not change any of the public part of the interface or the given student record type and you must implement all of the given public methods**. Your hashing function should use the primary key (the student ID) as input. Your database should be implemented as a chained hash table (see Figure 9.14 of the text), where each chain is an **ordered** list (ordered using the student's last name). When searching for a given last name, you should take advantage of the fact that the lists are ordered.

Input

We will provide you with a driver routine for this assignment (in file main.C). The main() routine presents a menu to the user with 5 options: Insert a record, Remove the record corresponding to the student whose ID

exist for the student ID or last name specified by the user, print out an error message. Your program should also print out an appropriate message indicating if a record was successfully inserted or deleted.

Your program should keep track of the number of key comparisons (probes) made during a "session". A session begins when the program is started and ends when the user selects the Exit option on the program menu. Before your program exits, it should print out the following statistics regarding the number of comparisons needed for searches in that session:

1. The maximum number of comparisons for a successful search using a student ID
2. The maximum number of comparisons for an unsuccessful search using a student ID
3. The average number of comparisons for a successful search using a student ID
4. The average number of comparisons for an unsuccessful search using a student ID
5. The maximum number of comparisons for a successful search using a student last name
6. The maximum number of comparisons for an unsuccessful search using a student last name
7. The average number of comparisons for a successful search using a student last name
8. The average number of comparisons for an unsuccessful search using a student last name

Error Checking

Your routines should detect the following errors:

a) Inserting a duplicate record. Note that a student is uniquely identified by student id. It is possible to have two students with the same name.
b) Entering a Student ID that is illegal. Each position in the Student ID string should be numeric character (0-9).

Notes

Note that the character at position *i* of a C++ string *S* can be accessed as *S[i]*. Also, the length of string *S* can be determined by *S.length()*. Comparison operators (= =, <=, etc) have been implemented for the string class.

Submitting your Assignment

The source code for this assignment should be submitted electronically as **Assignment #2**. Remember to turn in **all the files** (including header files and Makefile) that are needed to compile and run your program. The external documentation should be turned into your instructor (or the TA) in the first class after the due date. Remember that we do not accept late submissions. See online notes about electronic submission and documentation of your code. On the class web page, you can also see the grading criteria that will be used for all the assignments in this class.

Assignment Web Page

The files main.C and hash.H are available at http://mason.gmu.edu/~cs310/Assign2/