COP 3538 – Data Structures with OOP

Project 5 Fall 2011

Due: 30 Nov 2011 at 2pm – Note: No late assignments accepted after this date.

Please note that this assignment is shorter and will require a bit less effort.

Hash Tables

Using NetBeans 6.9.1 or later version, you are to write a Java program using OOP principles to accommodate the following functionality

Assignment #5

Objectives:

Provide student with exercises in hashing (randomizing) and collision processing

Provide student with experience in inserting, deleting, and changing node contents in a hash table

Functionality:

Given a sequential file, *States.Nicknames.txt* on my web page, you are to build a hash table based on team nickname ONLY.

Task 1: Build Hash Table: You are to build the hash table (array) in the order of appearance of the records in this input data file You are to use the hashing of strings algorithm (see book and lecture notes) for the algorithm to hash. You are to use separate chaining as your collision algorithm (see also textbook and my lecture notes). The resulting linked list of separate chaining does not have to be sorted.

Task2: Display Hash Table: Once the hash table is built from the input constrained by the hashing and collision algorithms cited above, you are to display the hash table and its links. Your format – always include nice headers, etc. – should appear in tabular (column) format: Hash table index (integer) in the left most column (starting with zero) followed by, say, five spaces, followed by the State name with spaces afterwards, followed by the state nickname stored in this home address. All three columns are to line up under a header that includes: Index State Name State NickName. Any items linked to this home address are to appear lined up and nicely spaced to the **right** of this first entry at this index, with five spaces in between any succeeding collision at that hash table address. Synonyms are to include state name and state nickname in that order. If a hash table entry, say item 16, has no occupants, then the 16 is still to appear in your output format under Index, but it will have the letters null.

The size of the hash table is to be 23. Thus output detail data must have 23 detail lines.

Task 3: Update the Hash Table: Using the *States.Nicknames.Update.txt*, you are to update your hash table. Inserts should be simple and use the same hashing and collision algorithms that were used to build the table. A ➔ Add; C ➔ Change; D ➔ Delete.

Deletions: These are to be flagged with a delete byte. Items are to be logically deleted and not physically deleted.

Changes: Here, you must locate the entry to be changed. Once found, you are to change the nickname.

Adds: Add entries to the hash table as you have already done.

Exception Handling: Dupe Adds are not allowed. You need not test for these. For a Change or Delete transaction not found, you will reflect this occurrence as described below.

Task 4: Display Updated Hash Table: Using the format described above, you are to display the updated hash table and its links. For those records that are logically deleted, but still physically present, you are to precede the nickname with an asterisk, as in: *Lone Star State.

For illegal Changes and Deletes, after displaying the updated hash table, skip a few lines and merely write the header: Illegal Transactions. Underneath, display the input transaction and an appropriate comment, such as Entry Not Found.

As usual, you have **PLENTY of time if you start right away and work slowly and methodically. Don't get caught by some surprise late in the game.**

Deliverable: Your zipped folder to me MUST include copies of the input files.

You are to zip all files in your P5 folder as expected and Send them to be via Blackboard using our standard naming conventions.

Grade Sheet – Program is worth 70 points.

Name: _____

Source Code – 20 points .....

I look for: Indentation
Internal comments
Scope terminators
Overall program structure

Outputs – 50 points .....

Accuracy and Format. All functionality present!
Skip lines in between displayed outputs as described above. .
Include headers / descriptors as you may feel appropriate, but they need to be respectable by my standards..

➔ If your program does not contain all the functionality required, you should not expect a passing grade on this program.

Please have an I/O class for your inputs, and a class for your States.nickname data. You will need a collection class that contains the hash table array and the methods used to process (build, update, display, etc.) the hash table.

Good luck and have fun!!!