CS20 – Section 093 – Fall 2019

Programming Assignment 7 – Due Sunday November 3 at 11:59PM

Hashing (60 points)

You are implementing an employee management system for a company with roughly 100 employees. You need a method of storing employee data. Each employee is uniquely represented by an SSN (a unique 9-digit number). Because you need to provide direct access to employee data by key, and because speed of processing has been determined to be the most important attribute of the system, you have chosen hashing as the storage method.

The employee data consists of the SSN, last name, and first name. Data records have the following format:

positions 0-8	SSN
positions 9-18	employee first name
positions 19-33	employee last name

Create an EmployeeData struct to represent employees. It has four members: SSN, first name, last name, and a linked list for the collision chain.

EmployeeData:

SSN	int
first name	string
last name	string
list	* to EmployeeData

In the struct, SSN is an integer; last and first names are strings. SSN comes in from the file as a string of character digits; convert this field into an integer before placing it into the data structure. The linked list contains nodes, each pointing to an EmployeeData node. Use the STL list to implement the chain for each hash table entry. (Optionally, you can use your own linked list ADT.)

Part 1. Populating the structure

In main, create a static array of EmployeeData elements, roughly 100 elements. This is the hash table. Choose an appropriate size for 100 employees based on the class discussion. Initialize the hash table to zeroes for the SSNs, and null strings for the names. At this point, the list header in each hash table entry won't be pointing to any nodes.

Create a test data file corresponding to the spec listed above. Read the file, creating one employee in the hash table for each record. When you read a record, run the key through a hash function and store the employee's data in that hash table entry. If the entry is already occupied, create a new EmployeeData node for the new employee, and add it to the linked list for that hash table entry. When you make up the data, be sure to include keys that cause collisions. Have at least 15 populated hash table entries, and have at least 2 of

those with 3 or more collisions. Include your test data file in the /src folder in your Eclipse project so that it gets submitted along with the rest of your project.

Part 2. Display the employees

On EOF, list the employees. Product a simple report that resembles the following:

```
hash table entry 2   employee 111111111 name Adam Adams
        there is no chain from this entry
hash table entry 3   employee 222222222 name Betty Bobbin
        chain - employee 333322224 name Charlie Charles
        chain - employee 454454554 name David Douglass
hash table entry 16   employee Erdle Earnheart
        there is no chain from this entry
end of list.  There are a total of 5 employees.
```

Part 3. Query the employee structure

Prompt the console user for an SSN. Display the data for that SSN, something like the following:

```
enter employee SSN
111111111
that is Adam Adams
enter employee SSN
454454554
that is David Douglass
0
Thank you and have a nice day
```

When the user enters zero for an SSN, they're done. Clean up thoroughly, and exit.

Design Decision: You could accomplish chaining in multiple ways. One thing to keep in mind is that in this assignment, you'll be adding elements to the hash table, and accessing them once they're in, but not changing them, nor deleting them after they've been added. It's probably easiest to use an STL list. But you can also use your own linked list ADT from assignment 4. You could also use an STL vector. You could create a stack for each entry in the hash table, and use the pointer in the hash table entry to point to the stack. You could implement that stack by using your stack ADT from assignment 5, or via an STL stack. Another item to consider is that if this was a "real life" implementation of employees, of course there would be additions, deletions, and changes to the data…employees get hired, leave the company, and change names, jobs, locations, etc.