Programming Assignment 7

CSCD300 Data Structure

Instructor: Dr. Bojian Xu
Eastern Washington University, Cheney, Washington

Due: 11:59pm, July 31, 2014 (Thursday)

Please follow these rules strictly:

1. Verbal discussions with classmates are encouraged, but each student must independently write his/her own work, without referring to anybody else's solution.
2. No one should give his/her code to anyone else.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.
4. Every source code file must have the author's name on the top.
5. All source code should be commented reasonably well.
6. Sharing any content of this assignment and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Hash table with chains using singly linked lists

In this programming assignment, we want to create a map of student profiles by implementing a hash table using singly linked list chains to solve the hashing collisions. Note that you need to implement your own singly linked list and CANNOT use Java API for the linked list. **We fix the hash table size to be `5`** in order to create many collisions so that you will have to solve the hash collisions by using the singly linked list. Accordingly, we will use the following hash function:

$$h(x) = (7x + 29) \mod 5$$

The design of your program

- You will need to have a singly linked list implementation, where each node has two data fields: `id, value`.
- You will need to create a hash table, which is an array of the singly linked list data type. Every array location is initialized to be the reference to an empty singly linked list object. Please note that it is a bad idea to use each array location to save the reference of the head node of the list. Think about the reason from the philosophy of the object oriented programming.
- All the elements that are hashed into an array location are going to be saved in the linked list whose reference is saved by that array location.

The specification of your program

1. Your program should be named as: `Test_HashChain.java`
2. The input and output of your program.

   The input is a pure ASCII text file, where each line is a student profile in the format of

   `id`, `name`

   where the `id` is a **4-digit** decimal number and the `name` is a pure English alphabet sequence. **You can assume all the `id`'s in the input file are distinct.**

   The output of your program is an interactive menu items, including `insert`, `delete`, `search`, and `print` operations, allowing the user to update/search/print your hash table.

An example

Suppose we supply to your program with a file named `data.txt` with the following content:

```
3255,Alice
6726,Bob
1237,Carol
0019,Doug
4308,Edward
```

Then, the command line you should type would be:

```
$java Test_HashChain data.txt
```

Then your program will read in all the elements, hash and save them in a hash table of **size 5**, where collisions are solved **by using singly linked lists**. Your program will then print the following menu items and options and wait for the user's choice.

```
Choose one of the following options.
===================================
1) insert/update a new student record
2) delete a student record
3) search for a student record
4) print all the student records
5) quit

Your choice:
```

1. If the user choose `1`, your program will prompt

   ```
   Input the student id:
   ```

   The user will then type the student's id. Your program will then prompt with

```
Input the student name:
```

The user will then type the student's name. If the provided student id exists, your program will print the following message and goes back to print the menu items again.

```
The student was exiting and the record has been updated.
```

If the student is not existing, your program will print the following message and goes back to print the menu items again.

```
The new student has been added successfully.
```

2. If the user chooses `2`, your program will prompt

```
Input the student id:
```

The user will then type the student's id. If the given student id exists, your program will print the following message and goes back to print the menu items again.

```
The student has been deleted successfully.
```

If the given student id does not exist, your program will print the following message and goes back to print the menu again.

```
No such a student.
```

3. If the user choose `3`, your program will prompt

```
Input the student id:
```

The user will then type in the student id. If the given student id exists, your program will print the following message and goes back to print the menu items again.

```
Student id:xxxx. Student name:xxxxx.
```

If the given student id does not exist, your program will print the following message and goes back to print the menu items again.

```
No such a student.
```

4. If the user chooses `4`, your program will print all the `(id,name)` out and then go back to print the menu items again. Print all the elements from the same linked list in one line, separated by a white space and each element being enclosed by a pair of parentheses `()`. By doing so, the user will be aware of the current structure of the hash table.

5. If the user chooses `5`, your program just exits.

Submission

- All your work files must be saved in one folder, named: **firstname_lastname_EWUID_cscd300_prog7**
  (1) We use the underline '_' not the dash '-'.
  (2) All letters are in the lower case including your name's initial letters.
  (3) If you have middle name(s), you don't have to put them into the submission's filename.
  (4) If your name contains the dash symbol '-', you can keep them.
- You need to include a pure ASCII text file in the above folder, which contains the description of your design and implementation of the hash table using singly linked list to solve the hash collisions.
- You then compress the above whole folder into a .zip file.
- Submit .zip file onto the Canvas system by the deadline.