COP3538 Project 5 – Hash Tables

Due Date: Friday, 12/8/2017 11:59 PM

Turn in:

Submit the zipped Eclipse program including at least Project5.java, HashTable.java, and States5.csv. The zip file should be named <your last name>_Project5.zip (for example, Liu_Project5.zip). The program should be well documented in the format of doc comments in Java. Detailed formats are found at http://www.oracle.com/technetwork/articles/java/index-137868.html.

Requirements:

Implement a HashTable class. The hash table will be created from States5.csv. Use the following class definition for a node in the table (put this inside your class and do not change it, except to add comments):

```
private class Node
{
   String stateName;
   int statePopulation;
   Node nextNode;

   public Node(String state, int population)
   {
      stateName = state;
      statePopulation = population;
   }
   public void printNode()
   {
      System.out.printf("%-25s%,10d\n", stateName,
statePopulation);
   }
}
```

The **HashTable** class must use **Separate Chaining** to deal with Collisions. For the array of linked lists in the HashTable class, you must use **singly linked list, NOT doubly linked list**! Moreover, insertions will happen at the beginning of the linked list, and searching and deletions will need to go through the list. You will need to implement this linked list class, so you must **NOT** use the *LinkedList* class provided by Java.

COP3538 Project 5 – Hash Tables

The class must use the following hash algorithm: add the Unicode values of all of the characters in the state name (including spaces) and modulus the result with **101** (this means that you need a hash array of 101 elements.) Unicode value of a character can be retrieved simply by using a type conversion from char to int. You must **NOT** use any hash function provided by Java libraries to compute hash values of state names.

(1). Create the **HashTable** class that implements the following public methods:

1. A no-arg constructor that creates an empty hash table.
2. The method: **public void insert(String state, int population)** that will insert a node into the proper position in the hash table based on state name.
3. The method: **public int find(String state)** that will search the table for the state of the given name and if found will return the population or -1 if not found.
4. The method: **public void delete(String state)** that fill find and delete the given state from the table.
5. The method: **public void display()** that will traverse the table and will print the hash table as follows:

```
1.  Empty
2.  Statename               populationvalue
    Statename               populationvalue
    Statename               populationvalue
3.  Empty
4.  Statename               populationvalue
5.  Empty
6.  Empty
. . .
101. Statename               populationvalue
```

6. The method: **public void printFreeAndCollisions()** that will print the number of empty spaces and the number of collisions in the table.

```
There are X spaces available and Y collisions in the hash table
```

(2). Create a class called Project5 that will

1. Read the States5.csv file of states and create a hash table by calling the **insert** method, and display the hash table by calling the **display** method.

2. Delete states **Vermont**, **California** and **South Carolina** from the hash table by calling the **delete** method.

3. Search for states **Florida**, **Rhode Island** and **California** by calling the **find** method. For the found state, print out its population.
4. Delete states **Kentucky**, **Minnesota**, **West Virginia** and **Ohio** from the hash table, and display the hash table by calling the **display** method.
5. Print the number of empty cells and the number of cells with collisions in the resulting hash table by calling the **printFreeAndCollisions** function.

Provide comments in this form for the **HashTable** classes:

Comments for the class:

```
/**
 * Detailed description of the class.
 *
 * @author <your name>
 * @version <date you last changed the class>
 */
```

Public method comments:

```
/**
 * Description of the purpose of the method, the meaning of the
 * input parameters (if any) and the meaning of the return values * (if any).
 *
 * @param  parameter  description of the parameter (one for each)
 * @return description of the return value
 */
```

Provide comments in this form for the **Project5** class.

```
/**
 * COP 3538: Project 5 – Hash Tables
 * <p>
 * Description of the class using as many lines as needed
 * with <p> between paragraphs. Including descriptions of the
 * input required and output generated.
 *
 * @author <your name>
 * @version <the date you last modified the program>
 */
public class Project5
{
```

Example Output:

```
COP3538 Project 5 – Xudong Liu
Hash Tables
Enter the file name: States5.csv

There were 50 state records read into the hash table.

Hash table content:

0.   Empty
1.   Empty
2.   Empty
3.   Kansas           2,893,957
4.   Oklahoma         3,850,568
5.   Empty
6.   California       38,332,521
…
40.  Vermont          626,630
     North Dakota     723,393
     Connecticut      3,596,080
…
100. Nebraska         1,868,516

Vermont has been deleted from hash table
California has been deleted from hash table
South Carolina has been deleted from hash table

Florida is found with a population of 19,552,860
Rhode Island is found with a population of 1,051,511
California is not found

Kentucky has been deleted from hash table
Minnesota has been deleted from hash table
West Virginia has been deleted from hash table
Ohio has been deleted from hash table

Hash table content:

0.   Empty
1.   Empty
2.   Empty
3.   Kansas           2,893,957
4.   Oklahoma         3,850,568
5.   Empty
6.   Empty
…
40.  North Dakota     723,393
     Connecticut      3,596,080
…
100. Nebraska         1,868,516

Hash table has X empty spaces and Y collisions.
```