**CSC220 Assignment11**

**Hash Tables**

The goal of this week’s assignment is:

1. Practice using hash tables

2. Learn about the importance of debugging

**Things you must do:**

1. There are many details in this assignment. Make sure you read the whole thing carefully before writing any code and closely follow this instruction.

2. You must complete your assignment individually.

3. Always remember Java is case sensitive.

4. Your file names, class names, and package name must match exactly as they are specified here.

5. Your project must include the methods you implemented in the lab.

**Things you must not do:**

1. You must not change the file names, class names, package names.

2. You must not change the signature of any of these methods (name, parameters, …). Just fill in the missing code inside them.

3. You must not create any different class.

**DO NOT** start your assignment unless you have all the features in the lab working!

**Part 1 – The problem description**

For this assignment you are required to finish implementing the following methods in **QuadraticProbingHashTable.java**:

1. **public QuadraticProbingHashTable( int size )**
2. **public int hash(int value, int tableSize)**
3. **public void insert( int x)**
4. **public void rehash()**
5. **public void remove( int x ) ←**
6. **public int find( int x ) ←**

**Part 2 –** remove **method**

The signature of this method should be:

public void remove( int x )

This function will remove an element from the hash table. Remember that you are not supposed to physically remove the values from the hash table. Instead, you are going to make them inactive by setting the isActive flag to be false.

**Part 3 –** find **method**

The signature of this method should be:

public int find( int x )

This function looks for a given value (called x) in the hash table and return its index, if present. If the value does not exist, your method is supposed to return -1. Note that this is NOT a simple indexing of the hash table due to potential of the presence of collision. So, you need to use the quadratic probing that you implemented as part of insert to help you find the position of value x (if present).

**Part 4 – Test your code**

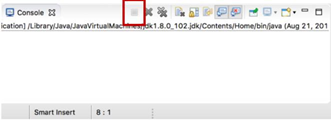
As usual you need to test the functionality of the methods you have implemented. A set of test cases has been provided for you as part of **QuadraticProbingHashTable.java**. Uncomment the assignment portion of the tests and run the main function. If you see any red text that says “TEST FAILED”, you need to debug your code.

How to debug your code?

1. Use the Eclipse debugger you learned about during the first lab.

2. If you see JavaStackOverflow, that means that you have an infinite recursive call and your recursive call is filling up the “call stack” (we talked about this concept in class). Go back and debug the method that is causing the problem.

3. Infinite loops! How would you know you have an infinite loop? As you should know from CSC120, if you have an infinite loop in your code, your code will not stop running. An easy way to inspect that in Eclipse is to look at your console window, if your code is done running the console should look like the following:



If the little square marked above is red and continues to stay red, that means your code has an infinite loop!

**Remarks**

* Make sure to submit your assignment by (re-)uploading your **Lab11** folder into your **csc220-cXXXX** folder by the deadline.
* **For all your assignments, please start early and seek help early (either from the instructor or the TAs).**