CSE 247 Data Structures and Algorithms Spring Semester 2019

Lab 7 Pre-Lab

Assigned: 3/6/2019 *Due Date: 3/19/2019*

Introduction

This pre-lab is designed to help prepare you to complete the programming component of Lab 8. Future coding labs will also include a pre-lab component.

Because the pre-lab section is designed to help you plan your implementation, it is strongly recommended for you to complete this section **before** writing any code. However, it is also natural for your understanding of the implementation requirements and your own strategy to evolve as you complete the code, so feel free to update your answers to the pre-lab section during/after the coding phase of the lab, up to the pre-lab deadline.

Questions

The programming component of this lab consists of implementing a hash table with collisions resolved by chaining. You'll be filling in the method stubs in the file `StringTable.java`. For each of the methods listed to be filled in (listed below), answer the following questions.

- If the method has a parameter (i.e. an input passed to the method), what does that parameter represent, in your own words? How will the parameter be used by the method? (An example of a parameter is that the method `insert` has one parameter `thing` of type `T`.)
- If the method has a non-`void` return value, what does it return, in your own words?

If the parameter/return is void, please state this fact in your answer. Here are the methods you will be filling in:

1. `public StringTable(int nBuckets);`
2. `public boolean insert(Record r);`
3. `public Record find(String key);`
4. `public void remove(String key);`
5. `private int toIndex(int hashcode);`

Here is a sample of an expected response for a method called `selectionSort` (not related to this assignment) that has the header

```
public int[] selectionSort(int[] arr);
```

and whose goal is to perform a selection sort on an array of ints.

Method: `public int[] selectionSort(int[] arr);`

- *Parameters*:
  `arr`: a 1-dimensional array of ints that stores the inputs to be sorted. The method will read this input array (multiple times) and write a sorted version of it to a newly allocated output array.
- *Return value*: a new array containing a sorted version of the input.

Once you finish your responses for the methods listed, answer the following additional questions:

6. What is the difference in meaning between the (public) `size` and the (private) `nBuckets` data members?
7. Which method(s) must you call, and in what order, to convert a String to an index into the array of hash buckets?
8. Where (i.e. in which function) will you allocate the linked list for each hash bucket?
9. Your hash calculations will involve the modulo operator '%'. Hashcodes in Java are signed integers; what does the Java modulo operator return when given a negative number? How might this impact your `toIndex()` function?