COMPX201-25A (HAM) & COMPX241-25A (HAM): Assignment Three

Hash Table & Testing

**Due: Friday 9th May, 11:59pm**

In this assignment, you will write a program which stores key/value pairs in a hash table. Your hash table must include a hash function that uses the “folding with strings” technique. You will be required to implement the functionality of a hash table using an appropriate data structure.

**Part One (40%):**

Define Java classes to implement a hash table of string key/value pairs following the specification given below:

1. **The Hash Table:** Define a class called StrHashTable in a file called StrHashTable.java. This class is to implement the following methods:

* insert(String k, String v) - adds the string key/value pair k,v to the hash table at the appropriate index. **Do not handle collisions, simply avoid inserting the new value.**
* delete(String k) - removes a key/value pair from the hash table, given the key k. Again, leave collisions for part two.
* hashFunction(String k) - using the folding method, create a hash function that returns the hash code for string k.
* rehash() - increases the size of the hash table (two-fold) when the load factor reaches 80% capacity.
* contains(String k) - returns true if string k is in the hash table, false otherwise.
* get(String k) - returns the item in the hash table, given the key k.
* isEmpty() - returns true if there is nothing in the hash table, false otherwise.
* count() - returns the number of items stored in the hash table.
* dump() - prints the contents of the hash table to the screen such that it matches the following format: index: key, value

0: key1, value1

1: key2, value2

2: key3, value3

3: …

1. **The Node:** define a class called Node for the nodes in your StrHashTable. It can either be an external class in a separate file called Node.java or an inner class of StrHashTable. It should have the following:

* A member variable to hold the string key.
* A member variable to hold the string value.
* A constructor that takes a key/value pair as two string arguments and copies them into the Node’s private member variable.

1. **Debugging:** Write a program class that creates one or more of your StrHashTable objects and tests that all your methods work as per the specification. You might, for example, write a program that reads words from a text file and puts them in your hash table counting the number of collisions that occur. Your program will not be marked but is for your own solution development process.

**Part Two (20%):**

Define a new class called StrHashTableCollisions in a separate file called StrHashTableCollisions.java that is a copy of your StrHashTable class except that each of the appropriate functions must be updated to handle collisions. To handle these collisions, your solution must use the separate chaining method as discussed in class. Ensure that you update the dump() function to print the full contents of the table. Lastly, update your debug program to check that your new class works as expected.

**Part Three (40%):**

Using JUnit as described in class, create a test file that will test the operations of your StrHashTableCollisions. For each test, consider the testing strategies discussed in class and ensure your test suite has adequate coverage of the functionality. Consider “edge” cases, like testing an empty hash table or cases where there are duplicate keys.

Add appropriate documentation to each of your tests to detail your reasons for adding the test to your test suite. It is useful for readability to group your tests for the same function together in the test suite.

Note that if you have implemented Part One correctly your tests for Part Three should pass. However, test quality is also important here, adding duplicate tests which cover the same functionality or code paths are meaningless and you will only receive marks for one of the duplicate tests. It may be useful to create a test plan to ensure that you cover the different parts of the functionality.

**Assessment**:

Completing Part One and Two can earn up to a C+ grade, but to be eligible for an A+ you must also implement Part Three. Your solution will be marked on the basis of how well it satisfies the specification, how well you have approached the problem (Solution Quality), how well-formatted and easy to read your code is (Code Quality), and whether each class and public method has at least some comment explaining what it does, what it’s for, and what any of its arguments are (i.e. documentation). Your code should compile and run as a console program from the command-line (i.e. no GUI or IDE).

**Submission**:

Create an empty directory (i.e. folder) using your student ID number as the directory name. Place copies of your source code (.java files) in this directory. If you wish to communicate with the marker any additional information then you may include a plain text README file, but nothing else (e.g. no compiled code (.class files) or IDE folders or files). Compress and upload this directory through the Moodle submission page for this assignment.