**CIS256 Project 7 Hash Tables**

**Part I: The HashTable**

Your textbook introduces a chained hash table class, ChainedTable. You are to download ChainedTable.java,

and complete the hash table implementation. In addition to those specified, add these two methods:

**public double estimateEfficiency()**

// Effect: determines the average number of comparisons (estimate) which must be made in a successful search

// of this hashtable (assumes a hash function which distributes well))

// Precondition: the table is not empty

// Returns: the value calculated using formula below

\*\* This value can be calculated using a formula from Knuth’s *The Art of Programming, VOL 3*

As follows:

1 + @/2 where @ = number of elements in the table

The size of the table’s array

Note: this is NOT integer division

**public double actualEfficiency ()**

// Effect: determines the average number of elements (actual) which were examined in a successful search

// of this hashtable

// Precondition: table is not empty

// Returns: the average number of values examined for all successful searches this hashtable has performed

\*\* This value must be determined by actually calculating the average number of comparisons which take place in a search

In your hash table.

**Part II: The Customer Class**

**\*Create a class Customer which implements the specification below:**

Instance data needed to be maintained by class Customer:

a Customer’s name (a string)

a Customer’s address (a string)

a Customer’s phone number (a Integer object) ( the hash table will use this as a key)

Constructors needed:

public Customer (String , String , Integer)

//constructor which accepts a name, address, and a phone number

Provide set and get methods for each piece of instance data.

Also provide a method which can be used when a String representation of a Customer object is needed.

String toString()

**Part III: Test Application**

*\* The Hash Function*

Design two hash functions for a seven digit phone number. You may assume that all phone numbers which will be hashed are from one or two exchanges, that is, the first three digits are either 574 or 416. Each of these hash functions will be used by to the hashIndex method (you will test both functions separately ).

You are to write a test application which:

1.) Creates a HashTable object with a capacity of 71

2.) stores 50 Customer objects in the hash table, Customer information can

be read from a file (will be easier)

3.) Outputs the best average search for this table, and the actual one

Run this application with BOTH of your hash functions, using the same input file, and see if you can determine which of your hash functions provides more evenly distributed indices. Summarize test results in your commenting of this file.

**Part IV:**  **Pizza Application (GUI is preferable)**

*Write an application* which can be used by Square Chair Pizza to store Customer info. This application will store all Customer objects in a hash table, and provide the following functionality:

\* a new Customer can be entered ; user first provides the Customer name, phone number and address

\* an existing Customer’s record can be retrieved ; user provides the phone number

* a Customer can be deleted ; user provides the phone number

**You are to submit your Customer.java, HashTable.java (include two hashCode methods, one can be commented out) your test program and your application.**