In today’s lab, we will be implementing our own Hash Table using separate chaining. You can use the ArrayList and LinkedList for this purpose. For simplicity, we assume that the key and value are strings:

**Task 1:** Write an Entry class with a String key and value and add a constructor:

private class Entry {

}

**Task 2:** We have declared the entries to be a List where each entry is a LinkedList. Initialize this list in the given constructor:

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

public class HashTable {

int SIZE = 0;

private int CAPACITY = 11;

private List<LinkedList<Entry>> entries;

// TODO: Initialize the entries. You may use an ArrayList for the entries where every item should be a Linked List. You need to initialize the ArrayList and a Linked List for each item of the ArrayList.

public HashTable() {

}

}

Next add the following methods:

// TODO: Generate an index to insert the key-value pair into the hash table at using the given key. You MUST USE String.hashCode() for this lab.

private int hash(String key) {

}

// TODO: Put the given key and value. Use hash function above to find the index. This tells you the location of the LinkedList where that key may reside. And then insert it in that Linked List ensuring that there are no duplicates. If the key is already present, then update the value. Otherwise, make a new entry to the Linked List. Size should be updated as well.

// If the load factor (Size/CAPACITY) would be 0.5 after you insert the new Entry, then rehash before inserting the new Entry. Make the size twice as much as it was (part of rehash), insert all entries into the new hash (part of rehash), and then insert the new Entry.

public void put(String key, String value) {

}

// TODO: get the value for a given key. Use hash above to find the index. This tells you the location of the LinkedList where that key may reside. If the key is not in the hash table, return null.

public String get(String key) {

}

// TODO: Remove the given key. Use hash above to find the index. This tells you the location of the LinkedList where that key may reside. Size should be updated as well.

public void remove(String key) {

}

// TODO: Double the capacity of this and re-insert each entry into the new HashTable.

public void rehash() {

}

**Task 3: Test your implementation.** Insert a number of items and output how many steps it took to insert and search for a given item.