~\OneDrive - VietNam National University - HCM INTERNATIONAL UNIVERSITY\Desktop\DSA\DSA LAB NEW\Lab 8 Graph\ITITSB22029_DoMinhDuy_Lab8\Problem 8\HashChainApp.java

```java
import java.io.*;

////////////////////////////////////////////////////////////////
class Link { // (could be other items)
    private int iData; // data item
    public Link next; // next link in list

    public Link(int it) // constructor
    {
        iData = it;
    }

    public int getKey() {
        return iData;
    }

    public void displayLink() // display this link
    {
        System.out.print(iData + " ");
    }
}
////////////////////////////////////////////////////////////////

class SortedList {
    private Link first; // ref to first list item

    public SortedList() // constructor
    {
        first = null;
    }

    public int insert(Link theLink) // insert link, in order
    {
        int key = theLink.getKey();
        Link previous = null;
        Link current = first;
        int probeCount = 1; // Start counting probes

        while (current != null && key > current.getKey()) {
            previous = current;
            current = current.next;
            probeCount++; // Increment probe count
        }

        if (previous == null)
            first = theLink; // Insert at beginning
        else
            previous.next = theLink;

        theLink.next = current;
        return probeCount; // Return number of probes
```

```
    }

    public void delete(int key) {
        Link previous = null;
        Link current = first;

        while (current != null && key != current.getKey()) {
            previous = current;
            current = current.next;
        }

        if (previous == null)
            first = first.next;
        else
            previous.next = current.next;
    }

    public Link find(int key) {
        Link current = first;
        int probeCount = 0;

        while (current != null && current.getKey() <= key) {
            probeCount++;
            if (current.getKey() == key) {
                System.out.println("Probes for finding key " + key + ": " + probeCount);
                return current;
            }
            current = current.next;
        }

        System.out.println("Probes for finding key " + key + ": " + probeCount);
        return null;
    }

    public void displayList() {
        System.out.print("List (first-->last): ");
        Link current = first;

        while (current != null) {
            current.displayLink();
            current = current.next;
        }

        System.out.println("");
    }
}
////////////////////////////////////////////////////////////////

class HashTable {
    private SortedList[] hashArray;
    private int arraySize;
    private int totalProbes; // Track total probes for inserts
    private int numInserts; // Track number of inserts

```

```
    public HashTable(int size) {
        arraySize = size;
        hashArray = new SortedList[arraySize];
        for (int j = 0; j < arraySize; j++) {
            hashArray[j] = new SortedList();
        }
        totalProbes = 0;
        numInserts = 0;
    }

    public void displayTable() {
        for (int j = 0; j < arraySize; j++) {
            System.out.print(j + ". ");
            hashArray[j].displayList();
        }
    }

    public int hashFunc(int key) {
        return key % arraySize;
    }

    public void insert(Link theLink) {
        int key = theLink.getKey();
        int hashVal = hashFunc(key);
        int probeCount = hashArray[hashVal].insert(theLink); // Insert and get probes
        totalProbes += probeCount;
        numInserts++;
        System.out.println("Inserted key: " + key + ", Probes: " + probeCount);
    }

    public void delete(int key) {
        int hashVal = hashFunc(key);
        hashArray[hashVal].delete(key);
    }

    public Link find(int key) {
        int hashVal = hashFunc(key);
        return hashArray[hashVal].find(key);
    }

    public double getAverageProbes() {
        return numInserts > 0 ? (double) totalProbes / numInserts : 0.0;
    }
}
////////////////////////////////////////////////////////////

class HashChainApp {
    public static void main(String[] args) throws IOException {
        int aKey;
        Link aDataItem;
        int size, n, keysPerCell = 100;

        System.out.print("Enter size of hash table: ");
        size = getInt();
```

```
160         System.out.print("Enter initial number of items: ");
161         n = getInt();
162 
163         HashTable theHashTable = new HashTable(size);
164 
165         System.out.println("Keys inserted during initial filling:");
166         for (int j = 0; j < n; j++) {
167             aKey = (int) (Math.random() * keysPerCell * size);
168             aDataItem = new Link(aKey);
169             System.out.print(aKey + " ");
170             theHashTable.insert(aDataItem);
171         }
172         System.out.println();
173 
174         double loadFactor = (double) n / size;
175         double avgProbes = theHashTable.getAverageProbes();
176         System.out.println("Load Factor: " + loadFactor);
177         System.out.println("Average Probe Length: " + avgProbes);
178 
179         while (true) {
180             System.out.print("Enter first letter of show, insert, delete, or find: ");
181             char choice = getChar();
182 
183             switch (choice) {
184                 case 's':
185                     theHashTable.displayTable();
186                     break;
187                 case 'i':
188                     System.out.print("Enter key value to insert: ");
189                     aKey = getInt();
190                     aDataItem = new Link(aKey);
191                     theHashTable.insert(aDataItem);
192                     break;
193                 case 'd':
194                     System.out.print("Enter key value to delete: ");
195                     aKey = getInt();
196                     theHashTable.delete(aKey);
197                     break;
198                 case 'f':
199                     System.out.print("Enter key value to find: ");
200                     aKey = getInt();
201                     aDataItem = theHashTable.find(aKey);
202                     if (aDataItem != null)
203                         System.out.println("Found " + aKey);
204                     else
205                         System.out.println("Could not find " + aKey);
206                     break;
207                 default:
208                     System.out.print("Invalid entry\n");
209             }
210         }
211     }
212 
213     public static String getString() throws IOException {
```

```
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        return br.readLine();
    }

    public static char getChar() throws IOException {
        return getString().charAt(0);
    }

    public static int getInt() throws IOException {
        return Integer.parseInt(getString());
    }
}
```