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

```java
import java.io.*;

class DataItem {
    private int iData;

    public DataItem(int ii) {
        iData = ii;
    }

    public int getKey() {
        return iData;
    }
}

class HashTable {
    private DataItem[] hashArray;
    private int arraySize;
    private DataItem nonItem;
    private int totalProbes;
    private int insertCount;

    HashTable(int size) {
        arraySize = size;
        hashArray = new DataItem[arraySize];
        nonItem = new DataItem(-1);
        totalProbes = 0;
        insertCount = 0;
    }

    public void displayTable() {
        System.out.print("Table: ");
        for (int j = 0; j < arraySize; j++) {
            if (hashArray[j] != null && hashArray[j].getKey() != -1)
                System.out.print(hashArray[j].getKey() + " ");
            else
                System.out.print("** ");
        }
        System.out.println("");
    }

    public int hashFunc1(int key) {
        return key % arraySize;
    }

    public int hashFunc2(int key) {
        return 5 - key % 5;
    }

    public void insert(int key, DataItem item) {
        int hashVal = hashFunc1(key);
        int stepSize = hashFunc2(key);
```

```
        int probeCount = 0;

        while (hashArray[hashVal] != null && hashArray[hashVal].getKey() != -1) {
            hashVal += stepSize;
            hashVal %= arraySize;
            probeCount++;
        }
        hashArray[hashVal] = item;
        probeCount++;
        totalProbes += probeCount;
        insertCount++;

        System.out.println("Inserted key " + key + " with hash value " + hashVal +
                ", step size " + stepSize + ", probe count " + probeCount);
    }

    public DataItem find(int key) {
        int hashVal = hashFunc1(key);
        int stepSize = hashFunc2(key);
        int probeCount = 0;

        while (hashArray[hashVal] != null) {
            if (hashArray[hashVal].getKey() == key) {
                probeCount++;
                System.out.println("Found key " + key + " at hash value " + hashVal +
                        ", step size " + stepSize + ", probe count " + probeCount);
                return hashArray[hashVal];
            }
            hashVal += stepSize;
            hashVal %= arraySize;
            probeCount++;
        }
        System.out.println("Key " + key + " not found after " + probeCount + " probes.");
        return null;
    }

    public void calculateAverageProbeLength() {
        if (insertCount > 0) {
            double average = (double) totalProbes / insertCount;
            System.out.println("Average probe length: " + average);
        }
    }
}

class HashDoubleApp {
    public static void main(String[] args) throws IOException {
        int aKey;
        DataItem aDataItem;
        int size, n;

        System.out.print("Enter size of hash table (preferably a prime number): ");
        size = getInt();
        System.out.print("Enter initial number of items: ");
        n = getInt();
```

```
106
107         HashTable theHashTable = new HashTable(size);
108
109         System.out.println("Initial Key Sequence:");
110         for (int j = 0; j < n; j++) {
111             aKey = (int) (Math.random() * 2 * size);
112             aDataItem = new DataItem(aKey);
113             System.out.print(aKey + " ");
114             theHashTable.insert(aKey, aDataItem);
115         }
116         System.out.println();
117
118         theHashTable.calculateAverageProbeLength();
119
120         while (true) {
121             System.out.print("Enter first letter of show, insert, delete, or find: ");
122             char choice = getChar();
123             switch (choice) {
124                 case 's':
125                     theHashTable.displayTable();
126                     break;
127                 case 'i':
128                     System.out.print("Enter key value to insert: ");
129                     aKey = getInt();
130                     aDataItem = new DataItem(aKey);
131                     theHashTable.insert(aKey, aDataItem);
132                     break;
133                 case 'f':
134                     System.out.print("Enter key value to find: ");
135                     aKey = getInt();
136                     theHashTable.find(aKey);
137                     break;
138                 default:
139                     System.out.print("Invalid entry\n");
140             }
141         }
142     }
143
144     public static String getString() throws IOException {
145         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
146         return br.readLine();
147     }
148
149     public static char getChar() throws IOException {
150         return getString().charAt(0);
151     }
152
153     public static int getInt() throws IOException {
154         return Integer.parseInt(getString());
155     }
156 }
```