***** PROJECT 2 Hardcopy (pdf file) *****

****** Cover page ******

Class: 323 MW
Name: Adewole Adeoshun
Project: Project 2
Project name: Hash Table implementation in Java
Language: Java
Due date: 9/18/2025, Thursday before midnight, 11:59pm
Submit date: 9/18/2025

Top level algorithm steps:

```
Step 0: open files from argv

Step 1: create hash table with dummy nodes

Step 2: read one op & data from inFile

Step 3: compute index = hash(data)

Step 4: if op = + → call hashInsert

Step 5: if op = - → call hashDelete

Step 6: if op = ? → call hashRetrieval

Step 7: else → illegal op → logFile

Step 8: repeat until end of file

Step 9: print final hash table to outFile1 and logFile

Step 10: close all files
```

Illustration:

Adewale Adeoshun

+ = insert, - = delete, ? = retrieve information, % = illegal

input:

+ Angela +Kenneth +Ross +Aparna +Mark %Mark +Mark
?Mary -Kevin ?Kevin ?Ross +Farhan ?Angela
?Kenneth ?Mohammed

Inserts are: Angela, Kenneth, Ross, Aparna, Mark, Mark
(duplicate ignored), Farhan

Hash each name using ord (decimal number)

1 Angela

ord(A) = 10, ord(n) = 110, ord(g) = 103, ord(e) = 101, ord(l) = 108
ord(a) = 97

Sum = 10 + 110 + 103 + 101 + 108 + 97 = 529

529 mod 5 = 4 ∴ goes to bucket 4

2 Kenneth

ord(K) = 75, ord(e) = 101, ord(n) = 110, ord(n) = 110, ord(e) = 4
ord(t) = 116, h = 104

Sum = 75 + 101 + 110 + 110 + 101 + 116 + 104 = 717

717 mod 5 = 2 $\longrightarrow$ bucket 2

3 Ross

R = 82, o = 111, s = 115, s = 115

Sum = 423
0423 mod 5 = 3

4 Aparcle
A = 10, p = 112, a = 97, r = 114, c = 101, r = 99, e = 1[illegible]
Sum = 634
634 mod 5 = 4

5 Mark
M = 77, a = 97, r = 114, k = 107
Sum = 395, 395 mod 5 = 0

6 Farhan : F = 70, a = 97, r = 114, h = 104, a = 97, n = 110
Sum = 592, mod = 592 mod 5 = 2

$$\begin{tikzcd}
\text{Bucket} & 0 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & 3 \arrow[d] & 4 \arrow[d] \\
\text{dummy} & \text{Mark} \arrow[d] & \text{Null} & \text{Kenneth} \arrow[d] & \text{Ross} \arrow[d] & \text{Angela} \arrow[d] \\
 & \text{Null} & & \text{Farhan} \arrow[d] & \text{Null} & \text{Aparcle} \arrow[d] \\
 & & & \text{Null} & & \text{Null}
\end{tikzcd}$$

Source code:

```
/*

Name: Adewole Adeoshun

Course: CSCI 323 (Mon/Wed)

Instructor: Tsaiyun Phillips

ID: 24081306

HashTable

*/

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.*;

// ======================

// Class for listNode

// ======================

class listNode {

String data; // The data stored in the node

listNode next; // Pointer to the next node

// Constructor: create a new node with given data

listNode(String d) {
```

```
data = d;
next = null;
}
// Print node in the required format
void printNode(BufferedWriter out) throws IOException {
if (next != null)
out.write("(" + data + ", " + next.data + ") -> ");
else
out.write("(" + data + ", NULL) -> ");
}
}
// ========================
// Class for HashTable
// ========================
class HashTable {
private listNode[] table; // Array of linked lists (buckets)
private int size; // Number of buckets
// Constructor: initialize table with dummy heads
HashTable(int size) {
this.size = size;
table = new listNode[size];
for (int i = 0; i < size; i++) {
table[i] = new listNode("dummy");
}
```

```
}

// Hash function: simple mod of string hashCode

private int hashIndex(String data) {

int sum = 0;

for (int i = 0; i < data.length(); i++) {

sum += (int) data.charAt(i);

}

return sum % size;

}

// Insert a node into the table

void hashInsert(String data, BufferedWriter logFile) throws IOException {

int index = hashIndex(data);

logFile.write("*** Calling hashInsert: data= " + data + "\n");

logFile.write("*** enter hashInsert method; index= " + index + " data= " + data + "\n");

// find spot to insert

listNode spot = table[index];

while (spot.next != null && spot.next.data.compareTo(data) < 0) {

spot = spot.next;

}

if (spot.next != null && spot.next.data.equals(data)) {

logFile.write("*** Warning, data is already in the database! ***\n");

logFile.write("*** Leaving hashInsert (...) ***\n");

return;

}
```

```
listNode newNode = new listNode(data);

newNode.next = spot.next;

spot.next = newNode;

logFile.write("*** Leaving hashInsert (...) ***\n");

}

// Delete a node from the table

void hashDelete(String data, BufferedWriter logFile) throws IOException {

int index = hashIndex(data);

logFile.write("*** Calling hashDelete: data= " + data + "\n");

logFile.write("*** Inside hashDelete method. Index= " + index + " data= " + data + "\n");

listNode spot = table[index];

while (spot.next != null && !spot.next.data.equals(data)) {

spot = spot.next;

}

if (spot.next == null) {

logFile.write("*** Warning, data is *not* in the hashTable! ***\n");

} else {

spot.next = spot.next.next; // delete the node

}

logFile.write("*** Leaving hashDelete () ***\n");

}

// Retrieval: check if data is in the table

void hashRetrieval(String data, BufferedWriter logFile, BufferedWriter outFile2) throws IOException {

int index = hashIndex(data);
```

```
logFile.write("*** Calling hashRetrieval: data= " + data + "\n");

logFile.write("*** Inside hashRetrieval. Index= " + index + " data= " + data + "\n");

listNode spot = table[index];

while (spot.next != null && !spot.next.data.equals(data)) {

spot = spot.next;

}

if (spot.next == null) {

outFile2.write("*** Warning, the record is *not* in the database! ***\n");

} else {

outFile2.write("Yes, the record is in the database!\n");

}

}

// Print the entire hash table

void printHashTable(BufferedWriter out) throws IOException {

for (int i = 0; i < size; i++) {

out.write("HashTable[" + i + "] -> ");

listNode spot = table[i];

while (spot != null) {

spot.printNode(out);

spot = spot.next;

}

out.write("NULL\n");

}

}
```

```
}

// ======================

// Main class

// ======================

public class AdeoshunA_Project2_Main {

public static void main(String[] args) throws IOException {

// Check if the right number of arguments are passed

if (args.length != 5) {

System.out.println("Usage: java AdeoshunA_Project2_Main <inFile> <bucketSize> <outFile1>
<outFile2> <logFile>");

return;

}

// Parse command-line arguments

String inFile = args[0];

int bucketSize = Integer.parseInt(args[1]);

String outFile1Name = args[2];

String outFile2Name = args[3];

String logFileName = args[4];

// Create readers and writers

BufferedReader reader = new BufferedReader(new FileReader(inFile));

BufferedWriter outFile1 = new BufferedWriter(new FileWriter(outFile1Name));

BufferedWriter outFile2 = new BufferedWriter(new FileWriter(outFile2Name));

BufferedWriter logFile = new BufferedWriter(new FileWriter(logFileName));

// Create the hash table

HashTable table = new HashTable(bucketSize);
```

```
String line;
while ((line = reader.readLine()) != null) {
String[] parts = line.split(" ");
String op = parts[0];
String data = parts.length > 1 ? parts[1] : "";
int index = data.isEmpty() ? -1 : (data.hashCode() & 0x7fffffff) % bucketSize;
logFile.write("In main(): op= " + op + " data= " + data + " index= " + index + "\n");
if (op.equals("+")) {
table.hashInsert(data, logFile);
} else if (op.equals("-")) {
table.hashDelete(data, logFile);
} else if (op.equals("?")) {
table.hashRetrieval(data, logFile, outFile2); // retrieval goes to outFile2
} else {
logFile.write("op is an illegal operation!\n");
}
}
// At the end, print final hash table to outFile1
outFile1.write("*** In main() below is the final hash Table ***\n");
table.printHashTable(outFile1);
// Also print final hash table to logFile
logFile.write("*** In main() below is the final hash Table ***\n");
table.printHashTable(logFile);
// Close all streams
```

```
reader.close();

outFile1.close();

outFile2.close();

logFile.close();

}

}
```

---

inFile:

*** below is HashTable_Data1.txt ***

```
+ Angela
+ Kenneth
+ Ross
+ Aparece
+ Mark
% Mark
+ Mark
? Mark
- Kevin
? Kevin
? Ross
+ Farhan
? Angela
```

? Kenneth

? Mohammed

*** below is HashTable_Data2.txt ***

+ William

+ Syed

+ Ross

+ Aparece

+ Mark

+ Angela

+ Kenneth

% Mark

+ Mark

? Mark

- Kevin

? Kevin

? Ross

+ Farhan

? Angela

? Kenneth

- Kenneth

+ Kenneth

+ Zachary

? Farhan

+ Zachary

- Zachary

# Zachary

* Zachary

+ Harry

- Aparece

? Aparece

+ Aparece

+ Murphy

? Murphy

+ Chen

- Asadbek

+ Ragib

% Ragib

? William

+ Ragib

- Venai

? Syed

+ Venai

+ Venai

+ Clevon

+ Benjamin

- Benjamin

+ Mohammed

? Benjamin

+ Aparece

? Clevon

+ Mohammed

- Mohammed

? Mohammed

---

outFile1:

*** below is outFile1_Data1.txt ***

```
*** In main() below is the final hash Table ***
HashTable[0] -> (dummy, Mark) -> (Mark, NULL) -> NULL
HashTable[1] -> (dummy, NULL) -> NULL
HashTable[2] -> (dummy, Farhan) -> (Farhan, Kenneth) -> (Kenneth, NULL) -> NULL
HashTable[3] -> (dummy, Ross) -> (Ross, NULL) -> NULL
HashTable[4] -> (dummy, Angela) -> (Angela, Aparece) -> (Aparece, NULL) -> NULL
```

*** below is outFile1_Data2.txt ***

```
Yes, the record is in the database!
*** Warning, the record is *not* in the database! ***
Yes, the record is in the database!
Yes, the record is in the database!
Yes, the record is in the database!
Yes, the record is in the database!
*** Warning, the record is *not* in the database! ***
```

Yes, the record is in the database!

Yes, the record is in the database!

Yes, the record is in the database!

**** Warning, the record is *not* in the database! ****

Yes, the record is in the database!

**** Warning, the record is *not* in the database! ****

---

outFile2:

```
*** below is outFile2_Data1.txt ***

*** In main() below is the final hash Table ***

HashTable[0] -> (dummy, NULL) -> NULL

HashTable[1] -> (dummy, Angela) -> (Angela, Harry) -> (Harry, Ragib) -> (Ragib, NULL) -> NULL

HashTable[2] -> (dummy, Kenneth) -> (Kenneth, NULL) -> NULL

HashTable[3] -> (dummy, NULL) -> NULL

HashTable[4] -> (dummy, Venai) -> (Venai, William) -> (William, NULL) -> NULL

HashTable[5] -> (dummy, Ross) -> (Ross, NULL) -> NULL

HashTable[6] -> (dummy, NULL) -> NULL

HashTable[7] -> (dummy, Aparece) -> (Aparece, Murphy) -> (Murphy, NULL) -> NULL

HashTable[8] -> (dummy, Chen) -> (Chen, NULL) -> NULL

HashTable[9] -> (dummy, Farhan) -> (Farhan, Syed) -> (Syed, NULL) -> NULL

HashTable[10] -> (dummy, Clevon) -> (Clevon, Mark) -> (Mark, NULL) -> NULL


*** below is outFile2_Data2.txt ***
```

Yes, the record is in the database!

**** Warning, the record is *not* in the database! ****

Yes, the record is in the database!

Yes, the record is in the database!

Yes, the record is in the database!

Yes, the record is in the database!

**** Warning, the record is *not* in the database! ****

Yes, the record is in the database!

Yes, the record is in the database!

Yes, the record is in the database!

**** Warning, the record is *not* in the database! ****

Yes, the record is in the database!

**** Warning, the record is *not* in the database! ****

---

logFile:

```
*** below is logFile_Data1.txt ***

In main(): op= + data= Angela index= 4

*** Calling hashInsert: data= Angela

*** enter hashInsert method; index= 4 data= Angela

*** Leaving hashInsert (...) ***

In main(): op= + data= Kenneth index= 1

*** Calling hashInsert: data= Kenneth

*** enter hashInsert method; index= 2 data= Kenneth
*** Leaving hashInsert (...) ***

In main(): op= + data= Ross index= 3

*** Calling hashInsert: data= Ross

*** enter hashInsert method; index= 3 data= Ross

*** Leaving hashInsert (...) ***

In main(): op= + data= Aparece index= 0

*** Calling hashInsert: data= Aparece

*** enter hashInsert method; index= 4 data= Aparece

*** Leaving hashInsert (...) ***

In main(): op= + data= Mark index= 0

*** Calling hashInsert: data= Mark

*** enter hashInsert method; index= 0 data= Mark

*** Leaving hashInsert (...) ***

In main(): op= % data= Mark index= 0

op is an illegal operation!

In main(): op= + data= Mark index= 0

*** Calling hashInsert: data= Mark

*** enter hashInsert method; index= 0 data= Mark

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= ? data= Mark index= 0

*** Calling hashRetrieval: data= Mark

*** Inside hashRetrieval. Index= 0 data= Mark

In main(): op= - data= Kevin index= 4
```

```
*** Calling hashDelete: data= Kevin

*** Inside hashDelete method. Index= 4 data= Kevin

*** Warning, data is *not* in the hashTable! ***

*** Leaving hashDelete () ***

In main(): op= ? data= Kevin index= 4

*** Calling hashRetrieval: data= Kevin

*** Inside hashRetrieval. Index= 4 data= Kevin

In main(): op= ? data= Ross index= 3

*** Calling hashRetrieval: data= Ross

*** Inside hashRetrieval. Index= 3 data= Ross

In main(): op= + data= Farhan index= 2

*** Calling hashInsert: data= Farhan

*** enter hashInsert method; index= 2 data= Farhan

*** Leaving hashInsert (...) ***

In main(): op= ? data= Angela index= 4

*** Calling hashRetrieval: data= Angela

*** Inside hashRetrieval. Index= 4 data= Angela

In main(): op= ? data= Kenneth index= 1

*** Calling hashRetrieval: data= Kenneth

*** Inside hashRetrieval. Index= 2 data= Kenneth

In main(): op= ? data= Mohammed index= 4

*** Calling hashRetrieval: data= Mohammed

*** Inside hashRetrieval. Index= 3 data= Mohammed

*** In main() below is the final hash Table ***
```

```
HashTable[0] -> (dummy, Mark) -> (Mark, NULL) -> NULL

HashTable[1] -> (dummy, NULL) -> NULL

HashTable[2] -> (dummy, Farhan) -> (Farhan, Kenneth) -> (Kenneth, NULL) -> NULL

HashTable[3] -> (dummy, Ross) -> (Ross, NULL) -> NULL

HashTable[4] -> (dummy, Angela) -> (Angela, Aparece) -> (Aparece, NULL) -> NULL
```

*** below is logFile_Data2.txt ***

```
In main(): op= + data= William index= 0

*** Calling hashInsert: data= William

*** enter hashInsert method; index= 4 data= William

*** Leaving hashInsert (...) ***

In main(): op= + data= Syed index= 4

*** Calling hashInsert: data= Syed

*** enter hashInsert method; index= 9 data= Syed

*** Leaving hashInsert (...) ***

In main(): op= + data= Ross index= 3

*** Calling hashInsert: data= Ross

*** enter hashInsert method; index= 5 data= Ross

*** Leaving hashInsert (...) ***

In main(): op= + data= Aparece index= 4

*** Calling hashInsert: data= Aparece

*** enter hashInsert method; index= 7 data= Aparece

*** Leaving hashInsert (...) ***

In main(): op= + data= Mark index= 3
```

```
*** Calling hashInsert: data= Mark

*** enter hashInsert method; index= 10 data= Mark

*** Leaving hashInsert (...) ***

In main(): op= + data= Angela index= 10

*** Calling hashInsert: data= Angela

*** enter hashInsert method; index= 1 data= Angela

*** Leaving hashInsert (...) ***

In main(): op= + data= Kenneth index= 9

*** Calling hashInsert: data= Kenneth

*** enter hashInsert method; index= 2 data= Kenneth

*** Leaving hashInsert (...) ***

In main(): op= % data= Mark index= 3

op is an illegal operation!

In main(): op= + data= Mark index= 3

*** Calling hashInsert: data= Mark

*** enter hashInsert method; index= 10 data= Mark

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= ? data= Mark index= 3

*** Calling hashRetrieval: data= Mark

*** Inside hashRetrieval. Index= 10 data= Mark

In main(): op= - data= Kevin index= 5

*** Calling hashDelete: data= Kevin

*** Inside hashDelete method. Index= 3 data= Kevin
*** Warning, data is *not* in the hashTable! ***

*** Leaving hashDelete () ***

In main(): op= ? data= Kevin index= 5

*** Calling hashRetrieval: data= Kevin

*** Inside hashRetrieval. Index= 3 data= Kevin

In main(): op= ? data= Ross index= 3

*** Calling hashRetrieval: data= Ross

*** Inside hashRetrieval. Index= 5 data= Ross

In main(): op= + data= Farhan index= 8

*** Calling hashInsert: data= Farhan

*** enter hashInsert method; index= 9 data= Farhan

*** Leaving hashInsert (...) ***

In main(): op= ? data= Angela index= 10

*** Calling hashRetrieval: data= Angela

*** Inside hashRetrieval. Index= 1 data= Angela

In main(): op= ? data= Kenneth index= 9

*** Calling hashRetrieval: data= Kenneth

*** Inside hashRetrieval. Index= 2 data= Kenneth

In main(): op= - data= Kenneth index= 9

*** Calling hashDelete: data= Kenneth

*** Inside hashDelete method. Index= 2 data= Kenneth

*** Leaving hashDelete () ***

In main(): op= + data= Kenneth index= 9

*** Calling hashInsert: data= Kenneth
*** enter hashInsert method; index= 2 data= Kenneth

*** Leaving hashInsert (...) ***

In main(): op= + data= Zachary index= 5

*** Calling hashInsert: data= Zachary

*** enter hashInsert method; index= 7 data= Zachary

*** Leaving hashInsert (...) ***

In main(): op= ? data= Farhan index= 8

*** Calling hashRetrieval: data= Farhan

*** Inside hashRetrieval. Index= 9 data= Farhan

In main(): op= + data= Zachary index= 5

*** Calling hashInsert: data= Zachary

*** enter hashInsert method; index= 7 data= Zachary

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= - data= Zachary index= 5

*** Calling hashDelete: data= Zachary

*** Inside hashDelete method. Index= 7 data= Zachary

*** Leaving hashDelete () ***

In main(): op= # data= Zachary index= 5

op is an illegal operation!

In main(): op= * data= Zachary index= 5

op is an illegal operation!

In main(): op= + data= Harry index= 10

*** Calling hashInsert: data= Harry
*** enter hashInsert method; index= 1 data= Harry

*** Leaving hashInsert (...) ***

In main(): op= - data= Aparece index= 4

*** Calling hashDelete: data= Aparece

*** Inside hashDelete method. Index= 7 data= Aparece

*** Leaving hashDelete () ***

In main(): op= ? data= Aparece index= 4

*** Calling hashRetrieval: data= Aparece

*** Inside hashRetrieval. Index= 7 data= Aparece

In main(): op= + data= Aparece index= 4

*** Calling hashInsert: data= Aparece

*** enter hashInsert method; index= 7 data= Aparece

*** Leaving hashInsert (...) ***

In main(): op= + data= Murphy index= 10

*** Calling hashInsert: data= Murphy

*** enter hashInsert method; index= 7 data= Murphy

*** Leaving hashInsert (...) ***

In main(): op= ? data= Murphy index= 10

*** Calling hashRetrieval: data= Murphy

*** Inside hashRetrieval. Index= 7 data= Murphy

In main(): op= + data= Chen index= 8

*** Calling hashInsert: data= Chen

*** enter hashInsert method; index= 8 data= Chen

*** Leaving hashInsert (...) ***
```

```
In main(): op= - data= Asadbek index= 10

*** Calling hashDelete: data= Asadbek

*** Inside hashDelete method. Index= 1 data= Asadbek

*** Warning, data is *not* in the hashTable! ***

*** Leaving hashDelete () ***

In main(): op= + data= Ragib index= 0

*** Calling hashInsert: data= Ragib

*** enter hashInsert method; index= 1 data= Ragib

*** Leaving hashInsert (...) ***

In main(): op= % data= Ragib index= 0

op is an illegal operation!

In main(): op= ? data= William index= 0

*** Calling hashRetrieval: data= William

*** Inside hashRetrieval. Index= 4 data= William

In main(): op= + data= Ragib index= 0

*** Calling hashInsert: data= Ragib

*** enter hashInsert method; index= 1 data= Ragib

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= - data= Venai index= 6

*** Calling hashDelete: data= Venai

*** Inside hashDelete method. Index= 4 data= Venai

*** Warning, data is *not* in the hashTable! ***

*** Leaving hashDelete () ***
```

```
In main(): op= ? data= Syed index= 4

*** Calling hashRetrieval: data= Syed

*** Inside hashRetrieval. Index= 9 data= Syed

In main(): op= + data= Venai index= 6

*** Calling hashInsert: data= Venai

*** enter hashInsert method; index= 4 data= Venai

*** Leaving hashInsert (...) ***

In main(): op= + data= Venai index= 6

*** Calling hashInsert: data= Venai

*** enter hashInsert method; index= 4 data= Venai

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= + data= Clevon index= 5

*** Calling hashInsert: data= Clevon

*** enter hashInsert method; index= 10 data= Clevon

*** Leaving hashInsert (...) ***

In main(): op= + data= Benjamin index= 4

*** Calling hashInsert: data= Benjamin

*** enter hashInsert method; index= 1 data= Benjamin

*** Leaving hashInsert (...) ***

In main(): op= - data= Benjamin index= 4

*** Calling hashDelete: data= Benjamin

*** Inside hashDelete method. Index= 1 data= Benjamin

*** Leaving hashDelete () ***
```

```
In main(): op= + data= Mohammed index= 7

*** Calling hashInsert: data= Mohammed

*** enter hashInsert method; index= 5 data= Mohammed

*** Leaving hashInsert (...) ***

In main(): op= ? data= Benjamin index= 4

*** Calling hashRetrieval: data= Benjamin

*** Inside hashRetrieval. Index= 1 data= Benjamin

In main(): op= + data= Aparece index= 4

*** Calling hashInsert: data= Aparece

*** enter hashInsert method; index= 7 data= Aparece

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= ? data= Clevon index= 5

*** Calling hashRetrieval: data= Clevon

*** Inside hashRetrieval. Index= 10 data= Clevon

In main(): op= + data= Mohammed index= 7

*** Calling hashInsert: data= Mohammed

*** enter hashInsert method; index= 5 data= Mohammed

*** Warning, data is already in the database! ***

*** Leaving hashInsert (...) ***

In main(): op= - data= Mohammed index= 7

*** Calling hashDelete: data= Mohammed

*** Inside hashDelete method. Index= 5 data= Mohammed

*** Leaving hashDelete () ***
```

```
In main(): op= ? data= Mohammed index= 7

*** Calling hashRetrieval: data= Mohammed

*** Inside hashRetrieval. Index= 5 data= Mohammed

*** In main() below is the final hash Table ***

HashTable[0] -> (dummy, NULL) -> NULL

HashTable[1] -> (dummy, Angela) -> (Angela, Harry) -> (Harry, Ragib) -> (Ragib, NULL) -> NULL

HashTable[2] -> (dummy, Kenneth) -> (Kenneth, NULL) -> NULL

HashTable[3] -> (dummy, NULL) -> NULL

HashTable[4] -> (dummy, Venai) -> (Venai, William) -> (William, NULL) -> NULL

HashTable[5] -> (dummy, Ross) -> (Ross, NULL) -> NULL

HashTable[6] -> (dummy, NULL) -> NULL

HashTable[7] -> (dummy, Aparece) -> (Aparece, Murphy) -> (Murphy, NULL) -> NULL

HashTable[8] -> (dummy, Chen) -> (Chen, NULL) -> NULL

HashTable[9] -> (dummy, Farhan) -> (Farhan, Syed) -> (Syed, NULL) -> NULL

HashTable[10] -> (dummy, Clevon) -> (Clevon, Mark) -> (Mark, NULL) -> NULL
```