Institute Of Technology, Nirma university

BRANCH :- **Computer Science Engineering**

PRACTICAL SUBMISSION

|*|*STUDENT INFO*|*|

Name :- **Pratik Kansara**

Roll No. :- **20BCE510**

Division :- **E4**

|*|*SUBJECT INFO*|*|

Subject :- **Advanced Data Structures**

Practical No. :- **10**

Practical – 10

AIM :- Hash tables are important data structure. However, hash tables are subject to collision. Implement a program with a collision resolution technique with Insert, delete and display operation

Code:

HashLinkNode.java

```
public class HashLinkNode {

    private int key;

    private int value;

    private HashLinkNode next;

    public HashLinkNode(int key, int value) {
        this.key = key;
        this.value = value;
        this.next = null;
    }

    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }

    public int getKey() {
        return key;
    }

    public HashLinkNode getNext() {
        return next;
    }

    public void setNext(HashLinkNode next) {
        this.next = next;
    }

}
```

Hashmap.java

```
public class Hashmap {

    private final static int TABLE_SIZE = 10;

    HashLinkNode[] table;

    public Hashmap() {
        table = new HashLinkNode[TABLE_SIZE];

        for (int i = 0; i < TABLE_SIZE; i++)
            table[i] = null;
    }

    public int get(int key) {
        int hash = (key % TABLE_SIZE);
        if (table[hash] == null)
            return -1;
        else {

            HashLinkNode entry = table[hash];
            while (entry != null && entry.getKey() != key)
                entry = entry.getNext();
            if (entry == null)
                return -1;
            else
                return entry.getValue();
        }
    }

    public void put(int key, int value) {

        int hash = (key % TABLE_SIZE);

        if (table[hash] == null)

            table[hash] = new HashLinkNode(key, value);

        else {
            HashLinkNode entry = table[hash];
            while (entry.getNext() != null && entry.getKey() != key)
                entry = entry.getNext();

            if (entry.getKey() == key)
                entry.setValue(value);
            else
                entry.setNext(new HashLinkNode(key, value));
        }
    }

    public void print() {
        for (int i = 0; i < table.length; i++) {
            HashLinkNode temp = table[i];
            System.out.print(i + " : ");
            while (temp != null) {
```

```
                System.out.print(temp.getKey() + " -> " + temp.getValue() + " ,
");
                temp = temp.getNext();
            }
            System.out.println("");
        }
    }

    public void remove(int key) {

        int hash = (key % TABLE_SIZE);

        if (table[hash] != null) {

            HashLinkNode prevEntry = null;

            HashLinkNode entry = table[hash];

            while (entry.getNext() != null && entry.getKey() != key) {
                prevEntry = entry;
                entry = entry.getNext();
            }

            if (entry.getKey() == key) {
                if (prevEntry == null)
                    table[hash] = entry.getNext();
                else
                    prevEntry.setNext(entry.getNext());

            }
        }
    }
}
```

Main.java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Hashmap map = new Hashmap();
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("1. Add Entry");
            System.out.println("2. Delete Entry");
            System.out.println("3. Search Element");
            System.out.println("4. Display Table");
            System.out.println("5. Exit");
            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Enter key : ");
                    int key = sc.nextInt();
                    System.out.println("Enter val : ");
                    int val = sc.nextInt();
                    map.put(key, val);
                    break;
                case 2:
                    System.out.println("Enter key : ");
                    int k = sc.nextInt();
                    map.remove(k);
                    break;
                case 3:
                    System.out.println("Enter key To Search : ");
                    int k1 = sc.nextInt();
                    System.out.println(map.get(k1));
                    break;
                case 4:
                    map.print();
                    break;
                case 5:
                    System.exit(0);
            }
        }
    }
}
```

OUTPUT

```
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
1
Enter key :
11
Enter val :
101
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
1
Enter key :
21
Enter val :
201
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
1
Enter key :
31
Enter val :
301
```

```
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
1
Enter key :
12
Enter val :
202
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
1
Enter key :
22
Enter val :
202
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
1
Enter key :
32
Enter val :
302
```

```
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
4
0 :
1 : 11 -> 101 , 21 -> 201 , 31 -> 301 ,
2 : 12 -> 202 , 22 -> 202 , 32 -> 302 ,
3 :
4 :
5 :
6 :
7 :
8 :
9 :
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
2
Enter key :
21
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
4
0 :
1 : 11 -> 101 , 31 -> 301 ,
2 : 12 -> 202 , 22 -> 202 , 32 -> 302 ,
3 :
4 :
5 :
```

```
6 :
7 :
8 :
9 :
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
2
Enter key :
22
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
4
0 :
1 : 11 -> 101 , 31 -> 301 ,
2 : 12 -> 202 , 32 -> 302 ,
3 :
4 :
5 :
6 :
7 :
3. Search Element
4. Display Table
5. Exit
3
Enter key To Search :
31
301
```

```
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
3
Enter key To Search :
12
202
1. Add Entry
2. Delete Entry
3. Search Element
4. Display Table
5. Exit
3
Enter key To Search :
45
-1
```