Name: Vijay Misal
Div: C Batch: C3
Roll No: 233073
PRN No: 22320079

# Practical No: 8

---

**Title:** Store data of students using hashing function for roll number and implement linear probing using chaining without replacement and chaining with replacement algorithm

**Code:**

```java
import java.util.*;

class Student {

    int rollNumber;

    public Student(int rollNumber) {

        this.rollNumber = rollNumber;
    }

    public int getRollNumber() {
        return rollNumber;
    }
}

class ChainingWithoutReplacement {
    private LinkedList<Student>[] hashTable;
    private int size;

    public ChainingWithoutReplacement(int size) {
        this.size = size;
        hashTable = new LinkedList[size];
        for (int i = 0; i < size; i++) {
            hashTable[i] = new LinkedList<>();
        }
    }

    public int hashFunction(int key) {
        return key % size;
    }

    public void insert(Student student) {
        int index = hashFunction(student.getRollNumber());
```

```
        hashTable[index].add(student);
    }

    public void display() {
        for (int i = 0; i < size; i++) {
            System.out.print("[" + i + "]");
            for (Student student : hashTable[i]) {
                System.out.print(" -> " + student.getRollNumber());
            }
            System.out.println();
        }
    }
}

class ChainingWithReplacement {
    private Student[] hashTable;
    private int size;

    public ChainingWithReplacement(int size) {
        this.size = size;
        hashTable = new Student[size];
    }

    public int hashFunction(int key) {
        return key % size;
    }

    public void insert(Student student) {
        int index = hashFunction(student.getRollNumber());
        if (hashTable[index] == null) {
            hashTable[index] = student;
        } else {
            for (int i = (index + 1) % size; i != index; i = (i + 1) %
size) {
                if (hashTable[i] == null) {
                    hashTable[i] = student;
                    return;
                }
            }
            System.out.println("HashTable is full. Unable to insert.");
        }
    }

    public void display() {
        for (int i = 0; i < size; i++) {
            if (hashTable[i] != null) {
                System.out.println("[" + i + "] " +
hashTable[i].getRollNumber());
```

```
            }
        }
    }
}

public class PR8 {
    public static void main(String[] args) {
        ChainingWithoutReplacement chainingWithoutReplacement = new
ChainingWithoutReplacement(10);
        chainingWithoutReplacement.insert(new Student(101));
        chainingWithoutReplacement.insert(new Student(102));
        chainingWithoutReplacement.insert(new Student(111));
        chainingWithoutReplacement.insert(new Student(110));
        chainingWithoutReplacement.insert(new Student(115));
        chainingWithoutReplacement.insert(new Student(120));
        System.out.println("Chaining Without Replacement");
        chainingWithoutReplacement.display();

        ChainingWithReplacement chainingWithReplacement = new
ChainingWithReplacement(10);
        chainingWithReplacement.insert(new Student(101));
        chainingWithReplacement.insert(new Student(102));
        chainingWithReplacement.insert(new Student(111));
        chainingWithReplacement.insert(new Student(110));
        chainingWithReplacement.insert(new Student(115));
        chainingWithReplacement.insert(new Student(120));
        chainingWithReplacement.insert(new Student(121));
        chainingWithReplacement.insert(new Student(122));

        System.out.println("Chaining With Replacement");
        chainingWithReplacement.display();
    }
}
```

**Output:**

Chaining Without Replacement

[0] ⟶ 110 ⟶ 120

[1] → 101 → 111

[2] → 102

[3]

[4]

[5] → 115

[6]

[7]

[8]

[9]

Chaining With Replacement

[0] 110

[1] 101

[2] 102

[3] 111

[4] 120

[5] 115

[6] 121

[7] 122