



## **Experiment 1.4**

**Aim:** Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.

**Objectives:** To understand doubly and circular linked list

**Input/Apparatus Used:** Doubly and circular Linked List is used.

### **Procedure/Algorithm:**

- Step1. Create the new node
- Step2. Set the new node's next to itself (circular)
- Step3. If the list is empty, return new node.
- Step4. Set our new node's next to the front. Step5. Set tail's next to our new node.
- Step6. Return the end of the list.

### **Sample Code:**

```
import java.util.*;
public class DAAexp4{

    public class Node {
        int data;
        Node next;
        Node prev;

        public Node(int data) {
            this.data = data;
            this.next = null;
            this.prev = null;
        }
    }

    public static Node head;
    public static Node tail;
    public static int size;

    // add
    public void addFirst(int data) {
        Node newNode = new Node(data);
        size++;
```



**Course Name: DAA Lab**

**Course Code: 21ITH-311/21CSH-311**

```
if (head == null) {
    head = tail = newNode;
    return;
}

newNode.next = head;
head.prev = newNode;
head = newNode;
}

public void addLast(int data){
    Node newNode = new Node(data);
    size++;
    if(head == null){
        head = tail = newNode;
        return;
    }

    tail.next = newNode;
    tail = newNode;
    tail.prev = newNode;
}

public int removeFirst(){
    if(head == null){
        System.out.println("DLL is empty");
        return Integer.MIN_VALUE;
    }
    if(size == 1){
        int val = head.data;
        head = tail = null;
        size--;
        return val;
    }
    int val = head.data;
    head = head.next;
    head.prev = null;
    size--;
    return val;
}

public int removeLast(){
    if(head == null){
        System.out.println("DLL is empty");
        return Integer.MIN_VALUE;
    }
    else if(size == 1){
        int val = tail.data;
```



**Course Name: DAA Lab**

**Course Code: 21ITH-311/21CSH-311**

```
        head = tail = null;
        size = 0;
        return val;
    }
    Node temp = head;
    for(int i=0; i<size-2; i++){
        temp = temp.next;
    }
    int val = tail.data;
    temp.next = null;
    temp.prev = tail;
    temp = tail;
    size--;
    return val;
}

// print
public void print(){
    Node temp = head;
    while(temp != null){
        System.out.print(temp.data + "<->");
        temp = temp.next;
    }
    System.out.println("null");
}
public static void main(String[] args) {
    DAAexp4 dll = new DAAexp4();
    dll.addLast(1);
    dll.addLast(2);
    dll.addLast(3);
    dll.addLast(4);
    dll.print();
    System.out.println(dll.size);

    dll.removeLast();
    dll.print();
    System.out.println(dll.size);
}
}
```

**Observations/Outcome :**



Course Name: DAA Lab

Course Code: 21ITH-311/21CSH-311

```
1<->2<->3<->4<->null
4
1<->2<->3<->null
3
PS C:\Users\NISHANT>
```

### Time Complexity:

Insertion at the Beginning:  $O(1)$

Insertion at the End:  $O(1)$

Insertion at a Given Position:  $O(n)$

Deletion at the Beginning:  $O(1)$

Deletion at the End:  $O(1)$

Deletion at a Given Position:  $O(n)$

Traversal:  $O(n)$