**Experiment 1.4**

**Aim:** *Code to perform operation on singly and doubly Linked list*

**Objectives:** *To perform insertion and deletion on singly and doubly Linked list*

**Input/Apparatus Used:** *VS CODE*

# Procedure/Algorithm:

**Insertion at the Beginning:**

1. *Create a new node with the given value.*
2. *Set the new node's next pointer to the current head.*
3. *Update the head pointer to point to the new node.*

**Insertion at the End:**

1. *Create a new node with the given value.*
2. *If the list is empty, set the head pointer to the new node.*
3. *Otherwise, traverse the list until you reach the last node.*
4. *Set the last node's next pointer to the new node.*

**Deletion at the Beginning:**

1. *If the list is empty, return.*
2. *Store the head node in a temporary variable.*
3. *Update the head pointer to point to the next node.*
4. *Delete the temporary variable (old head).*

**Deletion at the End:**

1. *If the list is empty, return.*
2. *If there's only one node, delete the node and set the head pointer to null.*
3. *Traverse the list until you reach the second-to-last node.*
4. *Set the second-to-last node's next pointer to null.*
5. *Delete the last node.*

# Code:

# *import java.util.\*;*

# *class LL {*

# *Node head;*

# *private int size;*

# *LL () {*

# *size = 0;*

# *}*

# *public class Node {*

# *String data;*

# *Node next;*

# *Node(String data) {*

# *this.data = data;*

# *this.next = null;*

# *size++;*

# *}*

# *}*

# *public void addFirst(String data) {*

# *Node newNode = new Node(data);*

# *newNode.next = head;*

# *head = newNode;*

# *}*

# *public void addLast(String data) {*

# *Node newNode = new Node(data);*

# *if(head == null) {*

# *head = newNode;*

# *return;*

# *}*

# *Node lastNode = head;*

# *while(lastNode.next != null) {*

# *lastNode = lastNode.next;*

# *}*

# *lastNode.next = newNode;*

# *}*

# *public void printList() {*

# *Node currNode = head;*

# *while(currNode != null) {*

# *System.out.print(currNode.data+" -> ");*

# *currNode = currNode.next;*

# *}*

# *System.out.println("null");*

# *}*

# *public void removeFirst() {*

# *if(head == null) {*

# *System.out.println("Empty List, nothing to delete");*

# *return;*

# *}*

# *head = this.head.next;*

# *size--;*

# *}*

# *public void removeLast() {*

# *if(head == null) {*

# *System.out.println("Empty List, nothing to delete");*

# *return;*

# *}*

# *size--;*

# *if(head.next == null) {*

# *head = null;*

# *return;*

# *}*

# *Node currNode = head;*

# *Node lastNode = head.next;*

# *while(lastNode.next != null) {*

# *currNode = currNode.next;*

# *lastNode = lastNode.next;*

# *}*

# *currNode.next = null;*

# *}*

# *public static void main(String args[]) {*

# *LL list = new LL();*

# *list.addLast("is");*

# *list.addLast("a");*

# *list.addLast("list");*

# *list.printList();*

# *list.addFirst("this");*

# *list.printList();*

# *list.removeFirst();*

# *list.printList();*

# *list.removeLast();*

# *list.printList();*

# *}*

# *}*

# Observations/Outcome :

# 

# Time Complexity:

# *addFirst: 0(1)*

# *addLast: 0(N)*

# *removeFirst: 0(1)*

# *removeLast: 0(n)*