Experiment 1.4

Student Name: Harsh Kumar **UID: 22BCS15754**
Branch: BE-CSE **Section/Group: 603_FL/B**
Semester: 5th **Date of Performance:08/08/24**
Subject Name: DAA **Subject Code: 22CSH -311**

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

2. **Objective:** To understand singly and doubly circular linked list.

3. **Algorithm:**
   1. **For Singly Linked list:**
      - Create an empty linked list with head = null.
      - Create a new node with the given data.
      - Set the next pointer of the new node to the current head.
      - Update the head to point to the new node.
      - Print a message indicating the insertion.
      - If the list is empty (head == null), print a message indicating the list is empty.
      - If the list has only one node, print the data of the head node, set head = null, and return.
      - Traverse the list to find the second-to-last node.
      - Print the data of the last node.
      - Set the next pointer of the second-to-last node to null (removing the last node).
      - Initialize a pointer current to head.
      - Traverse the list, printing the data of each node until the end (current == null).
      - Print a newline after displaying all elements.

   2. **For Doubly Circular Linked list:**
      - Create an empty circular doubly linked list with head = null.
      - Create a new node with the given data.
      - If the list is empty (head == null), set head = newNode.
      - Otherwise, link the new node between the last node (head.prev) and the current head.
      - Update the last node's next and the current head's prev to point to the new node.
      - Update head to point to the new node.

- Print a message indicating the insertion.
- If the list is empty (head == null), print a message indicating the list is empty.
- If the list has only one node (head.next == head):
- Print the data of the head node.
- Set head = null.
- Otherwise, find the last node (head.prev).
- Update the second-to-last node's next to point to head.
- Update head.prev to point to the second-to-last node.
- Print a message indicating the deletion.
- If the list is empty (head == null), print a message indicating the list is empty.
- Otherwise, start at the head node and traverse the list, printing each node's data.
- Continue until you loop back to the head node.
- Print a newline after displaying all elements.

4. Implementation/Code:

```
class Node {
    int data;
    Node next;
Node(int data) {
this.data = data;
this.next = null;
    }
} class SinglyLinkedList {
private Node head;
public SinglyLinkedList() {
head = null;
        }
public void insertAtBeginning(int data) {

Node newNode = new

Node(data);

newNode.next = head;

head = newNode;

System.out.println("Element "+data+" is inserted.");
        }
public void deleteFromEnd() {
```

```
if (head == null) {
System.out.println("Linked list is
empty.");
return;
        }
if (head.next == null) {
System.out.println("Element " + head.data + " is deleted.");
head = null;
return;
        }
Node current = head;            while
(current.next.next != null) {
current = current.next;
        }
System.out.println("Element " + current.next.data + " is deleted.");
current.next = null;
    }
public void display() {
Node current = head;
while (current != null) {
System.out.print(current.data + " ");
current = current.next;
        }
System.out.println();
    }
}
public class fourth {
public static void main(String[] args) {
SinglyLinkedList list = new SinglyLinkedList();
list.insertAtBeginning(10);
list.insertAtBeginning(20);
list.insertAtBeginning(30);
list.insertAtBeginning(40);
list.display();
list.deleteFromEnd();
list.display();
}
}
```

```
//Doubly Circular Linked list class
Node {
int data;
Node next;
Node prev;
Node(int data) {
this.data = data;
this.next = this;
this.prev = this;
    }
}
class CircularDoublyLinkedList {
private Node head;     public
CircularDoublyLinkedList() {
        head = null;
    }
public void insertAtBeginning(int data) {
Node newNode = new Node(data);
if (head == null) {
head = newNode;
        } else {
            Node last = head.prev;
newNode.next = head;
newNode.prev = last;
last.next = newNode;
head.prev = newNode;
            head = newNode;
        }
System.out.println("Element " + data + " is inserted.");
    }
public void deleteFromEnd() {
if (head == null) {
            System.out.println("Linked list is empty.");
return;
        }
if (head.next == head) {
```

```
            System.out.println("Element " + head.data + " is deleted.");
head = null;
            return;
        }
Node last = head.prev;
last.prev.next = head;
head.prev = last.prev;
System.out.println("Element " + last.data + " is deleted.");
    }
public void display() {
if (head == null) {
            System.out.println("Linked list is empty.");
return;
        }
Node current = head;
do {
            System.out.print(current.data + " ");
current = current.next;         }
while (current != head);
System.out.println();
    }
}
public class fourth {
    public static void main(String[] args) {
        CircularDoublyLinkedList list = new CircularDoublyLinkedList();
list.insertAtBeginning(15);
list.insertAtBeginning(27);
list.insertAtBeginning(35);
list.insertAtBeginning(48);
        list.display();
list.deleteFromEnd();
list.display();
    }
}
```

5. Output:

Singly Linked list:

```
Element 10 is inserted.
Element 20 is inserted.
Element 30 is inserted.
Element 40 is inserted.
40 30 20 10
Element 10 is deleted.
40 30 20
```

Doubly Circular Linked list:

```
Element 15 is inserted.
Element 27 is inserted.
Element 35 is inserted.
Element 48 is inserted.
48 35 27 15
Element 15 is deleted.
48 35 27
```

6. Time Complexity:

- **Singly Linked List:**
  - Insert at Beginning: O(1). ○ Delete from End: O(n).
  - Display: O(n).
- **Doubly Circular Linked List:**
  - Insert at Beginning: O(1). ○ Delete from End: O(1).
  - Display: O(n).

7. Learning Outcome:

1. I have learnt about Singly Linkedlist.
2. I have learnt the concept of insertion and deletion in Singly Linked list.
3. I have learnt about Doubly Circular Linked list.
4. I have learnt the concept of insertion and deletion in Doubly Circular Linked list.
5. I have learnt how to analyze the time complexity of algorithms.