Experiment 4

Student Name: Rajiv Paul **UID:**20BCS1812
Branch: CSE **Section/Group:** 702 A
Semester: 5th **Date of Performance:** 01/09/2022
Subject Name: DAA Lab **Subject Code:** 20-CSP-312

1. Aim/Overview of the practical:

(i) Code for inserting and removing elements at the start and end of a doubly and circular linked list.

2. Task to be done/ Which logistics used:

To write code for inserting and removing elements at the start and end of a doubly and circular linked list.

3. Algorithm/Flowchart (For programming based labs):

4. Steps for experiment/practical/Code:

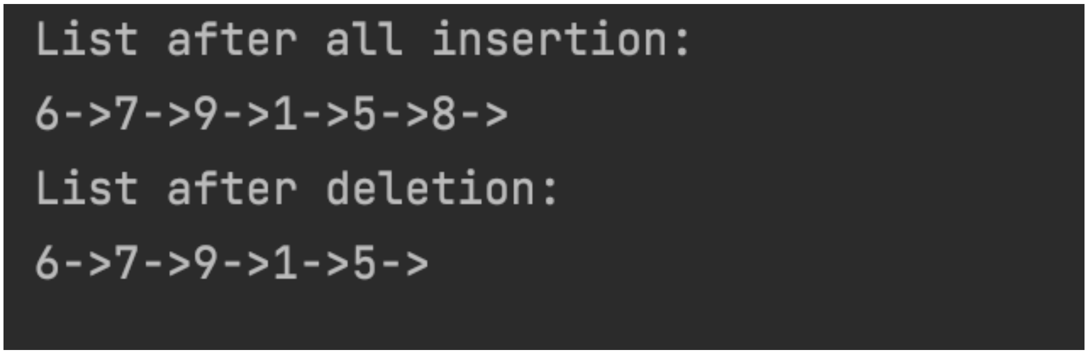
```
package com.DAA;

    class DoublyLinkedList {


    Node head;

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

        Node(int d) {
            data = d;
        }
    }


    public void insertFront(int data) {

        Node newNode = new Node(data);


        newNode.next = head;


        newNode.prev = null;


        if (head != null)
            head.prev = newNode;


        head = newNode;
    }
```

```
public void insertAfter(Node prev_node, int data) {


    if (prev_node == null) {
        System.out.println("previous node cannot be null");
        return;
    }


    Node new_node = new Node(data);


    new_node.next = prev_node.next;


    prev_node.next = new_node;


    new_node.prev = prev_node;


    if (new_node.next != null)
        new_node.next.prev = new_node;
}


void insertEnd(int data) {

    Node new_node = new Node(data);


    Node temp = head;


    new_node.next = null;


    if (head == null) {
        new_node.prev = null;
        head = new_node;
        return;
    }
    while (temp.next != null)
        temp = temp.next;


    temp.next = new_node;


    new_node.prev = temp;
}


void deleteNode(Node del_node) {


    if (head == null || del_node == null) {
        return;
    }


    if (head == del_node) {
        head = del_node.next;
    }


    if (del_node.next != null) {
        del_node.next.prev = del_node.prev;
    }


    if (del_node.prev != null) {
        del_node.prev.next = del_node.next;
    }

}


public void printlist(Node node) {
    Node last = null;
    while (node != null) {
        System.out.print(node.data + "->");
        last = node;
        node = node.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        DoublyLinkedList doubly_ll = new DoublyLinkedList();

        doubly_ll.insertEnd(5);
        doubly_ll.insertFront(1);
        doubly_ll.insertFront(6);
        doubly_ll.insertEnd(8);



        doubly_ll.insertAfter(doubly_ll.head, 7);


        doubly_ll.insertAfter(doubly_ll.head.next, 9);

        System.out.println("List after all insertion:");

        doubly_ll.printlist(doubly_ll.head);


        doubly_ll.deleteNode(doubly_ll.head.next.next.next.next.next);

        System.out.println("List after deletion:");

        doubly_ll.printlist(doubly_ll.head);
    }
}
```

5. Observations/Discussions/ Complexity Analysis:

Time complexity is O(n).

6. Result/Output/Writing Summary:



```
List after all insertion:
6->7->9->1->5->8->
List after deletion:
6->7->9->1->5->
```

1. Aim/Overview of the practical:

(ii) Using templates, write code to push and pop elements, check Isempty and Isfull, and return the top element in stacks.

2. Task to be done/ Which logistics used:

To write code to push and pop elements, check Isempty and Isfull, and return the top element in stacks.

3. Algorithm/Flowchart (For programming based labs):

4. Steps for experiment/practical/Code:

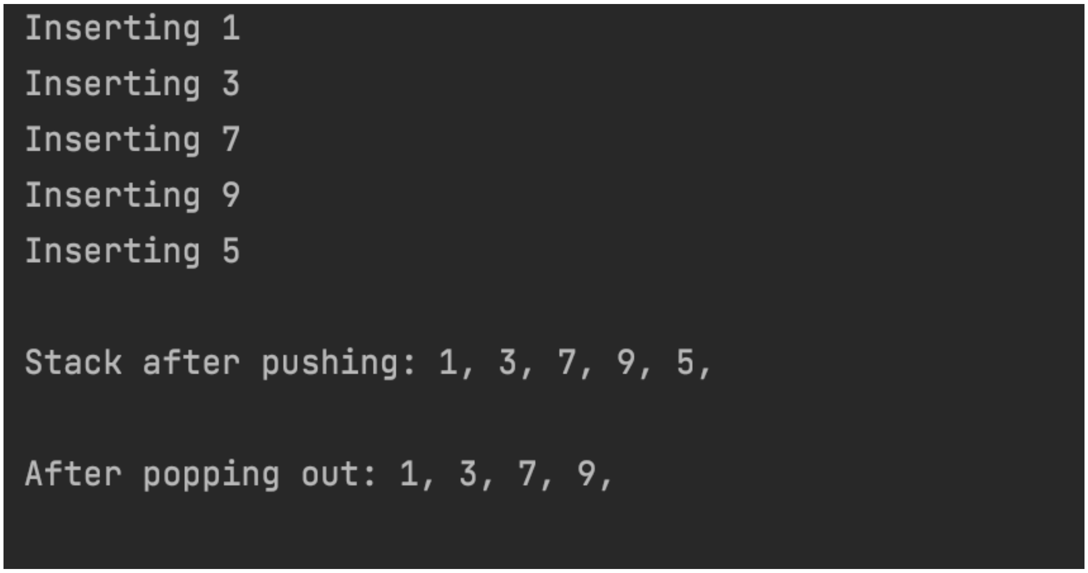
```
package com.DAA;

class Stack {

    private int arr[];

    private int top;

    private int cap;

    Stack(int size) {

        arr = new int[size];
        cap = size;
        top = -1;
    }


    public void push(int x) {
        if (isFull()) {
            System.out.println("Stack OverFlow");


            System.exit(1);
        }


        System.out.println("Inserting " + x);
        arr[++top] = x;
    }


    public int pop() {


        if (isEmpty()) {
            System.out.println("Stack is Empty");

            System.exit(1);
        }


        return arr[top--];
    }


    public int getSize() {
        return top + 1;
    }


    public Boolean isEmpty() {
        return top == -1;
    }
```

```
    public Boolean isFull() {
        return top == cap - 1;
    }


    public void printStack() {
        for (int i = 0; i <= top; i++) {
            System.out.print(arr[i] + ", ");
        }
    }
    public static class DAA_exp1_4_ii {
        public static void main(String[] args) {
            Stack stack = new Stack(5);

            stack.push(1);
            stack.push(3);
            stack.push(7);
            stack.push(9);
            stack.push(5);

            System.out.print("\nStack after pushing: ");
            stack.printStack();


            stack.pop();
            System.out.print("\n\nAfter popping out:");
            stack.printStack();

        }

    }

}
```

5. Observations/Discussions/ Complexity Analysis:

Time complexity is O(1).

6. Result/Output/Writing Summary:



```
Inserting 1
Inserting 3
Inserting 7
Inserting 9
Inserting 5

Stack after pushing: 1, 3, 7, 9, 5,

After popping out: 1, 3, 7, 9,
```

Learning outcomes (What I have learnt):

1. **Learnt about doubly linked list and its implementation.**
2. **Learnt how to insert from start and end.**
3. **Learnt how to delete from start and end.**
4. **Learnt about push and pop in stack.**
5.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			