**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);

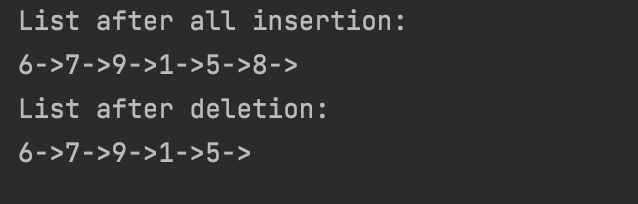
}

}

**5. Observations/Discussions/ Complexity Analysis:**

Time complexity is O(n).

**6. Result/Output/Writing Summary:**

****

**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();

}

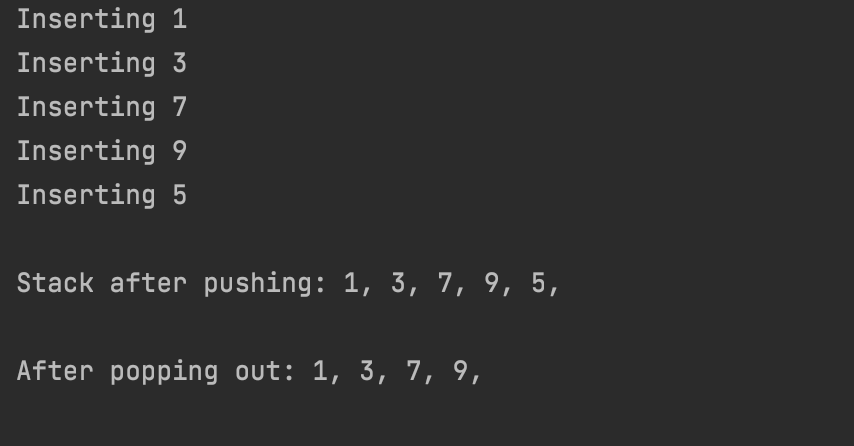
}

}

**5. Observations/Discussions/ Complexity Analysis:**

Time complexity is O(1).

**6. Result/Output/Writing Summary:**

****

**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. |  |  |  |
|  |  |  |  |