Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

\_\_\_\_**04**\_\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Write a program to create a doubly linked list and pertform:  \*traversing  \* Insertion  \*deletion |
|  |  |
|  |  |
|  |  |
|  |  |

Submitted On

26/10/2022

(Date: DD/MM/YY)

**Task No. 1 : Write a program to create a doubly linked list and pertform**

**\*Traversing**

**\* Insertion**

**\*Deletion**

**Solution:**

**Main Class:**

DLL dll = new DLL();

dll.head = new Node(1);

Node second = new Node(2);

Node third = new Node(3);

Node fourth = new Node(4);

Node fifth = new Node(5);

dll.head.next = second;

second.next = third;

third.next = fourth;

fourth.next = fifth;

fifth.prev = fourth;

fourth.prev = third;

third.prev = second;

second.prev = dll.head;

dll.printList(second);

dll.reverse(third);

dll.append(6);

dll.push(7);

dll.center(dll.head.next, 8);

Console.WriteLine("Created DLL is: ");

dll.printList(dll.head);

dll.deleteNode(dll.head.next);

Console.WriteLine("\nList after Deleting middle node: ");

dll.printList(dll.head);

Node head = null;

Console.WriteLine("\n\n\* \* \* \* AFTER DELETION \* \* \* \*");

dll.deleteNode(fourth);

dll.printList(dll.head);

Console.WriteLine("\n\n\* \* \* \* AFTER DELETION (AT GIVEN POSITION) \* \* \* \*");

dll.deleteNodeAtGivenPos(dll.head, 2);

dll.printList(dll.head);

**DLL Class:**

public Node head;

public void printList(Node node){

Node last = null;

Console.WriteLine("Traversal in Forward Direction");

while (node != null){

Console.Write(node.data + " ");

last = node;

node = node.next;}

Console.WriteLine("\nTraversal in Reverse Direction");

while (last != null){

Console.Write(last.data + " ");

last = last.prev;}}

public void reverse(Node node)

{

Console.WriteLine("\n Node value in Reverse Direction");

Node last = node;

while (last != null)

{

Console.Write(last.data + " ");

last = last.prev;}}

public void push(int new\_data) {

Node new\_Node = new Node(new\_data);

new\_Node.next = head;

new\_Node.prev = null;

if (head != null) {

head.prev = new\_Node;

head = new\_Node;}}

public void center(Node prev\_Node, int center\_data){

Node center\_Node = new Node(center\_data);

if (prev\_Node == null){

Console.WriteLine("The given previous node cannot be NULL ");

return;}

center\_Node.next = prev\_Node.next;

prev\_Node.next = center\_Node;

center\_Node.prev = prev\_Node;

if (center\_Node.next != null){

center\_Node.next.prev = center\_Node;} }

public void append(int new\_data){

Node new\_node = new Node(new\_data);

Node last = head;

new\_node.next = null;

if (head == null){

new\_node.prev = null;

head = new\_node;

return;}

while (last.next != null)

last = last.next;

last.next = new\_node;

new\_node.prev = last;}

public void deleteNode(Node del) {

if (head == null || del == null)

return;

if (head == del)

head = del.next;

if (del.next != null)

del.next.prev = del.prev;

if (del.prev != null)

del.prev.next = del.next;

return;}

public void deleteNodeAtGivenPos(Node head, int n){

if (head == null || n <= 0)

return;

Node current = head;

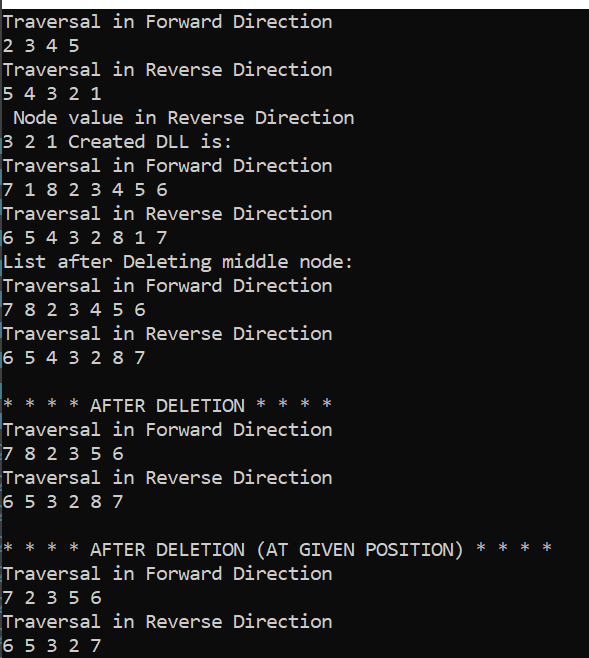
int i;

for (i = 1; current != null && i < n; i++)

current = current.next;

if (current == null)

return;

deleteNode(current);

**Node Class:**

public int data;

public Node next;

public Node prev;

public Node(int d){

data = d;}

**Output:**