Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

\_\_\_\_**03**\_\_\_\_\_

LIST OF TASKS

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

Submitted On:

\_\_\_\_18/10/2022\_\_\_\_

(Date: DD/MM/YY)

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

**\*Traversing**

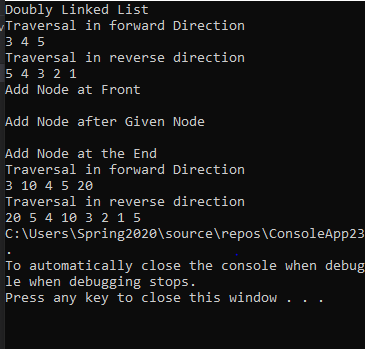
**\* Insertion**

**\*Deletion**

**Solution:**

* **Traversing:**

**Main Class:**



**DDL Class:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

internal class DLL

{

public Node head;

public void printList(Node node)

{

Node last=null;

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

while (node != null)

{

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

last = node;

node = node.next;

}

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

while (last != null)

{

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

last = last.prev;

}

}

public void reverse(Node node)

{

Console.WriteLine("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.Write("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;

}

static 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;

// delete the node pointed to by 'current'

deleteNode(head, current);

}

}

}

**Node Class:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

internal class Node

{

public int data;

public Node prev;

public Node next;

public Node(int d)

{

data = d;

}

}

}

Main Class:

using System;

namespace ConsoleApp3

{

class Program

{

static void Main(string[] args)

{

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.push(1);

dll.append(4);

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

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

dll.printList(dll.head);

dll.deleteNode(dll.head.next);

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

dll.printList(dll.head);

Node head = null;

// Create the doubly linked list:

// 2<->2<->10<->8<->4<->2<->5<->2

head = push(head, 2);

head = push(head, 5);

head = push(head, 4);

head = push(head, 8);

head = push(head, 10);

Console.WriteLine("Doubly linked list before deletion:");

printList(head);

int n = 2;

// delete node at the given position 'n'

deleteNodeAtGivenPos(head, n);

Console.WriteLine("Doubly linked list after deletion:");

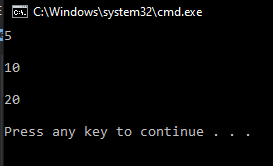
printList(head);

}

}

}

**Output:**



* **Insertion**

**Solution:**

**Main Class:**

LinkedList list = new LinkedList();

list.head = new Node(5);

Node second = new Node(10);

Node third = new Node(20);

list.head.next = second;

second.next = third;

list.printList();

list.append(6);

list.push(7);

list.push(1);

list.insertAfter(list.head.next, 8);

Console.Write("\n\nCreated Linked list is: \n\n");

list.printList();

**LinkedList Class:**

public Node head;

public void printList(){

Node n=head;

while(n!=null){

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

n = n.next;}}

public void push(int new\_data){

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

new\_node.next = head;

head = new\_node;}

public void insertAfter(Node prev\_node, int new\_data){

if (prev\_node == null){

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

return;}

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

new\_node.next = prev\_node.next;

prev\_node.next = new\_node;}

public void append(int new\_data){

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

if (head == null){

head = new Node(new\_data);

return;}

new\_node.next = null;

Node last = head;

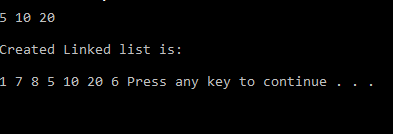
while (last.next != null)

last = last.next;

last.next = new\_node;

return;}

**Output:**



* **Deletion**

**Solution:**

**Main Class:**

LinkedList list = new LinkedList();

list.head = new Node(5);

Node second = new Node(10);

Node third = new Node(20);

list.head.next = second;

second.next = third;

list.printList();

list.append(6);

list.push(7);

list.push(1);

list.insertAfter(list.head.next, 8);

Console.Write("\n\nCreated Linked list is: \n\n");

list.printList();

list.deleteNode(1);

**LinkedList Class:**

public void deleteNode(int key){

Node temp = head, prev = null;

if (temp != null &&

temp.data == key){

head = temp.next;

return;}

while (temp != null &&

temp.data != key){

prev = temp;

temp = temp.next;}

if (temp == null)

return;

prev.next = temp.next;}

**Output:**

