MCA Semester 1	Subject : Advanced Data Structures Lab
Name : Mukund Gangurde	Topic: Applications of Linked Lists DLL & Polynomial
Roll No. : MCA2511	Date : 14-11-2025

1) Demonstrate the working of a Doublylinked list with operations to insert, delete, search, display and count the number of nodes.

Code:

```
import java.util.Scanner;

class DNode
{
	int data;
	DNode prev;
	DNode next;

	DNode(int d)
	{
		data = d;
		prev = null;
		next = null;
	}
}//end of DNode

//DList
class DList
{
	DNode head;
	DNode tail;

	public DList()
	{
		head = null;
		tail = null;
	}

	//Insert
	public void Insert(int x)
	{
		//1. Make a new node t
		DNode t =new DNode(x);
```

```
        //2. First node in DLL
        if(head == null)
        {
                head = t;
                tail = t;
        }
        else    //3. Link t at the end of the DLL
        {
                tail.next = t;
                t.prev = tail;
                tail = t;
        }
}//end of Insert

//Delete
public void Delete(int x)
{
        //1. Search for x
        DNode tmp = head;
        DNode p = null;
        boolean flag = false;

        while(tmp!=null)
        {
                if(tmp.data == x)
                {
                        flag = true;
                        break;
                }
                p = tmp;
                tmp = tmp.next;
        }

        //2. Unsuccessful Search
        if(flag == false)
        {
                System.out.println(x + "is not found");
                return;
        }

        //3. Successful Search
        if(tmp == head && tmp == tail)          //A. Single node Deletion
        {
                head = null;
                tail = null;
        }
```

```
        else if (tmp == head)           //B. Head Node Deletion
        {
                head = tmp.next;
                head.prev = null;
        }
        else if (tmp == tail)           //C. Tail node Deletion
        {
                tail = p;
                tail.next = null;
        }
        else                            //D. Any other node deletion
        {
                p.next = tmp.next;
                tmp.next.prev = p;
        }
}//end of Delete

//Search
public void Search(int x)
{
        DNode tmp = head;
        boolean flag = false;

        while(tmp!=null)
        {
                if(tmp.data == x)
                {
                        flag = true;
                        break;
                }
                tmp = tmp.next;
        }
        if(flag==true)
        {
                System.out.println(x+ " is found");
        }
        else
        {
                System.out.println(x+ " is not found");
        }
}//end of Search

//Count
public void Count()
{
        DNode tmp = head;
```

```
		int cnt = 0;

		while(tmp!=null)
		{
			cnt++;
			tmp = tmp.next;
		}
		System.out.println("There are "+ cnt + " nodes");
	}//end of Count

	//Display
	public void Display()
	{
		DNode tmp = head;
		System.out.print("From head: ");
		while(tmp!=null)
		{
			System.out.print(tmp.data + "<->");
			tmp = tmp.next;
		}
		System.out.print("NULL");
		System.out.println();

		tmp = tail;
		System.out.print("From tail: ");
		while(tmp!=null)
		{
			System.out.print(tmp.data + "<->");
			tmp = tmp.prev;
		}
		System.out.print("NULL\n");
	}//end of Display


}//end of DList

class DLL
{
	public static void main(String[] args)
	{
		int ch;
		DList d = new DList();
		Scanner sc = new Scanner(System.in);

		do
		{
```

```
System.out.println("\n*****Doubly Linked List******");
System.out.println("1. Insert a new node in DLL");
System.out.println("2. Delete a node in DLL");
System.out.println("3. Search for a node in DLL");
System.out.println("4. Count No. of Nodes in DLL");
System.out.println("5. Display Nodes in DLL");
System.out.println("6. Exit out");
System.out.println();

System.out.print("Enter your choice: ");
ch = sc.nextInt();

switch(ch)
{
        case 1:
                System.out.print("Enter a value: ");
                int x = sc.nextInt();
                d.Insert(x);
                d.Display();
        break;

        case 2:
                d.Display();
                System.out.print("Enter a value: ");
                x = sc.nextInt();
                d.Delete(x);
                d.Display();
        break;

        case 3:
                System.out.print("Enter a value: ");
                x = sc.nextInt();
                d.Search(x);
        break;

        case 4:
                d.Count();
        break;

        case 5:
                d.Display();
        break;

        case 6:
                System.out.println("Exiting -------   :");
        break;
```

```
                        default:
                                System.out.println("Incorrect .... \n");
                        break;
                }


                } while (ch != 6);
        }//end of psvm
}// end of DLL
```

Output:

Data:

```
Enter your choice: 1
Enter a value: 10
From head: 101<->50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->101<->NULL

*****Doubly Linked List******
1. Insert a new node in DLL
2. Delete a node in DLL
3. Search for a node in DLL
4. Count No. of Nodes in DLL
5. Display Nodes in DLL
6. Exit out

Enter your choice: 5
From head: 101<->50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->101<->NULL
```

Node Not Found

```
*****Doubly Linked List******
1. Insert a new node in DLL
2. Delete a node in DLL
3. Search for a node in DLL
4. Count No. of Nodes in DLL
5. Display Nodes in DLL
6. Exit out

Enter your choice: 2
From head: 101<->50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->101<->NULL
Enter a value: 123
123is not found
From head: 101<->50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->101<->NULL
```

Head Node

```
*****Doubly Linked List******
1. Insert a new node in DLL
2. Delete a node in DLL
3. Search for a node in DLL
4. Count No. of Nodes in DLL
5. Display Nodes in DLL
6. Exit out

Enter your choice: 2
From head: 101<->50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->101<->NULL
Enter a value: 101
From head: 50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->NULL
```

Tail Node

```
*****Doubly Linked List******
1. Insert a new node in DLL
2. Delete a node in DLL
3. Search for a node in DLL
4. Count No. of Nodes in DLL
5. Display Nodes in DLL
6. Exit out

Enter your choice: 2
From head: 50<->60<->30<->80<->60<->20<->10<->NULL
From tail: 10<->20<->60<->80<->30<->60<->50<->NULL
Enter a value: 10
From head: 50<->60<->30<->80<->60<->20<->NULL
From tail: 20<->60<->80<->30<->60<->50<->NULL
```

Any other Node

```
*****Doubly Linked List******
1. Insert a new node in DLL
2. Delete a node in DLL
3. Search for a node in DLL
4. Count No. of Nodes in DLL
5. Display Nodes in DLL
6. Exit out

Enter your choice: 2
From head: 50<->60<->30<->80<->60<->20<->NULL
From tail: 20<->60<->80<->30<->60<->50<->NULL
Enter a value: 80
From head: 50<->60<->30<->60<->20<->NULL
From tail: 20<->60<->30<->60<->50<->NULL
```

2. Polynomial Insertion & Display.

Code:

```
import java.util.Scanner;

//Node Template
class Node
{
	int coeff;
	int exp;
	Node next;

	public Node (int c, int a)
	{
		coeff = c;
		exp = a;
		next = null;
	}//end of Node Constructor
}//end of class Node

//List template for Polynomial
class Polynomial
{
	Node head;

	public Polynomial()
	{
		head = null;
	}

	//Insert
	void Insert(int c, int e)
	{
		//Make a new Node
		Node t = new Node(c,e);

		//First Node in the list
		if(head == null)
		{
			head = t;
			return;
		}

		//Traverse and check for order in exponent
		Node tmp = head;
		Node prev = null;
```

```
        //Find location for t
        while(tmp!=null && t.exp < tmp.exp)
        {
                prev = tmp;
                tmp = tmp.next;
        }

        //Insert at right location
        if(tmp == head)                 //Head Node Insertion
        {
                t.next = tmp;
                head = t;
        }
        else if (tmp == null)   //Tail Node Insertion
        {
                prev.next = t;
        }
        else                            //Any other node Deletion
        {
                prev.next = t;
                t.next = tmp;
        }
}//end of Insert

//Display
public void Display()
{
        //CHeck for Empty List
        if(head == null)
        {
                System.out.println("Empty List");
                return;
        }

        //Traverse and check for order in exponent
        Node tmp = head;

        //Find location for t
        while(tmp!=null)
        {
                System.out.print(tmp.coeff + "x^" + tmp.exp);
                if(tmp.next!=null)
                {
                        System.out.print(" + ");
                }
```

```
                tmp = tmp.next;
            }
        }//end of Display

        //Sum / Addition

}//end of Polynomial

class PolAdd
{
        public static void main(String[] args)
        {
            Scanner sc = new Scanner(System.in);
            Polynomial p1 = new Polynomial();
            Polynomial p2 = new Polynomial();

            System.out.println("Enter the number of terms in Polynomial 1: ");
            int n1 = sc.nextInt();
            System.out.println("Enter Polynomial 1: (coeff,exp) ");

            for(int i = 1; i<=n1 ; i++)
            {
                int c = sc.nextInt();
                int e = sc.nextInt();
                p1.Insert(c,e);
            }

            System.out.print("Polynomial 1: ");
            p1.Display();

        }//end of psvm
}//end of PolAdd
```

Output:

```
A:\MCA2511\DS_LAB>java PolAdd
Enter the number of terms in Polynomial 1:
5
Enter Polynomial 1: (coeff,exp)
12 6
13 8
23 2
45 4
6 3
Polynomial 1:
13x^8 + 12x^6 + 45x^4 + 6x^3 + 23x^2
```