MCA Semester 1	Subject : Advanced Data Structures Lab
Name : Mukund Gangurde	Topic: Applications of Linked Lists 1) Polynomial Addition 2) Stack as a List
Roll No. : MCA2511	Date : 17-11-2025

1) Polynomial Addition.

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(Math.abs(tmp.coeff) + "x^" + tmp.exp);
                if(tmp.next!=null)
                {
                        if(tmp.next.coeff > 0)
                                System.out.print(" + ");
                        else
                                System.out.print(" - ");
                }
                tmp = tmp.next;
        }
}//end of Display

//Sum / Addition
static Polynomial addPolynomial (Polynomial p1, Polynomial p2)
{
        Polynomial res = new Polynomial();
        Node a = p1.head;
        Node b = p2.head;

        while(a!=null && b!=null)
        {
                if(a.exp > b.exp)
                {
                        res.Insert(a.coeff, a.exp);
                        a = a.next;
                }
                else if(a.exp < b.exp)
                {
                        res.Insert(b.coeff,b.exp);
                        b = b.next;
                }
                else
                {
                        int tot = a.coeff + b.coeff;
                        res.Insert(tot,a.exp);
                        a = a.next;
                        b = b.next;
                }
        }//end of while a AND b

        //Add remaining terms from a
        while(a!=null)
```

```
                {
                        res.Insert(a.coeff,a.exp);
                        a = a.next;
                }

                //Add remaining terms from a
                while(b!=null)
                {
                        res.Insert(b.coeff,b.exp);
                        b = b.next;
                }

                return res;
        }
}//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.print("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();

                System.out.println();
                System.out.print("Enter the number of terms in Polynomial 2: ");
                int n2 = sc.nextInt();
                System.out.println("Enter Polynomial 2: (coeff,exp) ");

                for(int i = 1; i<=n2 ; i++)
                {
                        int c = sc.nextInt();
```

```
                int e = sc.nextInt();
                p2.Insert(c,e);
            }

            System.out.print("Polynomial 2: ");
            p2.Display();

            System.out.println();
            System.out.println();
            System.out.println("Sum of the Polynomial: ");
            Polynomial result = new Polynomial();
            result = result.addPolynomial(p1,p2);

            result.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)
4 5
3 4
3 3
3 2
3 1
Polynomial 1: 4x^5 +
3x^4 +
3x^3 +
3x^2 +
3x^1
Enter the number of terms in Polynomial 2: 3
Enter Polynomial 2: (coeff,exp)
2 4
1 3
2 0
Polynomial 2: 2x^4 +
1x^3 +
2x^0

Sum of the Polynomial:
4x^5 +
5x^4 +
4x^3 +
3x^2 +
3x^1 +
2x^0
A:\MCA2511\DS_LAB>
```

2) Stack as a List.

Code:

```
import java.util.Scanner;

//Node template
class Node
{
	int data;
	Node next;

	public Node(int d)
	{
		data = d;
		next = null;
	}
}//end of Node

//Lsit based Stack Template
class ListStack
{
	Node tos;

	public ListStack()
	{
		tos = null;
	}//end of ListStack Constructor

	//Push
	public void Push(int x)
	{
		Node t = new Node(x);

		if(tos == null)
		{
			tos = t;
		}
		else
		{
			t.next = tos;
			tos = t;
		}
		System.out.println(t.data + " Pushed into stack");
	}//end of Push

	//Pop
	public void Pop()
	{
		Node tmp = tos;

		if(tmp == null)
		{
```

```
                System.out.println("Stack Underflowed!");
                return;
            }
            System.out.println(tos.data + " Popped from stack");
            tos = tmp.next;
        }//end of Pop

        //Peek
        public void Peek()
        {
            Node tmp = tos;

            if(tmp == null)
            {
                System.out.println("Stack Underflowed!");
            }
            else
            {
                System.out.println("Element at the top is: " + tmp.data);
            }
        }//end of Peek

        //Display
        public void Display()
        {
            Node tmp = tos;

            if(tmp == null)
            {
                System.out.println("Stack Underflowed!");
                return;
            }
            System.out.print("Stack contains ");
            while(tmp != null)
            {
                System.out.print(tmp.data+ " ");
                tmp = tmp.next;
            }
        }//end of Display
}//end of ListStack

//Main
class LStack
{
        public static void main(String[] args)
        {
            ListStack s= new ListStack();
            Scanner sc = new Scanner(System.in);
            int ch, x;

            do
            {
```

```
                System.out.println("\n\nList Based Stack\n");

                System.out.println("1. Push an Element in the Stack");
                System.out.println("2. Pop an Element from the Stack");
                System.out.println("3. Peek at the Stack");
                System.out.println("4. Display the Stack");
                System.out.println("5. Exit");

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

                switch(ch)
                {
                        case 1:
                                System.out.println("Enter a value: ");
                                x = sc.nextInt();
                                s.Push(x);
                        break;

                        case 2:
                                s.Pop();
                        break;

                        case 3:
                                s.Peek();
                        break;

                        case 4:
                                s.Display();
                        break;

                        case 5:
                                System.out.println("Exiting....");
                        break;

                        default:
                                System.out.println("Incorrect Choice: ");
                        break;
                }
        } while(ch!=5);
    }//end of psvm
}//end of LStack
```

Output:

```
A:\MCA2511\DS_LAB>javac 16LStack.java

A:\MCA2511\DS_LAB>java LStack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :1
Enter a value:
12
12 Pushed into stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :1
Enter a value:
13
13 Pushed into stack


List Based Stack

1. Push an Element in the Stack
```

```
1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :1
Enter a value:
56
56 Pushed into stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :1
Enter a value:
13
13 Pushed into stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :4
Stack contains 13 56 13 12
```

```
1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :2
13 Popped from stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :2
56 Popped from stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :4
Stack contains 13 12

List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
```

```
4. Display the Stack
5. Exit
Enter your choice :3
Element at the top is: 13


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :2
13 Popped from stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :2
12 Popped from stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
```

```
List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :2
12 Popped from stack


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :2
Stack Underflowed!


List Based Stack

1. Push an Element in the Stack
2. Pop an Element from the Stack
3. Peek at the Stack
4. Display the Stack
5. Exit
Enter your choice :5
Exiting....

A:\MCA2511\DS_LAB>
```