CS1102:
DATA STRUCTURES

COURSE-LAB-FILE

SUBMITTED BY:

KONDROLLA DINESH REDDY

(2020BTechCSE040)

FACULTY GUIDE:

MR. DEVENDRA BHAVSAR

Institute of Engineering and Technology (IET)
JK Lakshmipat University Jaipur

NOV 2021

Question-1:

1. **Write a program to search an element in the Array using Linear Search.**

Solution:

```
package LabFile;

import java.util.*;
public class LinearSearch {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                int[] array = new int[10];
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter the Size of Array");
                int size = sc.nextInt();
                System.out.println("Enter the Vales in the Array");
                for(int i=0;i<size;i++) {
                        array[i] = sc.nextInt();
                }
                for(int i=0;i<size;i++) {
                        System.out.print(array[i] + " ");
                }
                System.out.println();
                System.out.println("Enter the Value which you want to search");
                int val = sc.nextInt();

                int i=0,count=0;
                while(i<size) {
                        if(array[i] == val) {
                                count++;
                        }
                        i++;
                }

                if(count == 0) {
                        System.out.println("THE VALUE IS NOT IN THE GIVEN ARRAY");
                }
                else {
                        System.out.println("WE FOUND THE VALUE - " + val + " -> " +count + "
TIMES");
                }
        }

}
```

Question-2:

Write a program to implement Binary Search in an Array.

Solution:

```
package LabFile;

import java.util.*;
public class BinarySearchArr {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the Size of Array :");
        int size = sc.nextInt();
        int[] arr = new int[size];
        System.out.println("Enter the values of Array (Ascending Order) :");
        for(int i=0; i<arr.length;i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println();
        System.out.println("The Array : ");
        for(int i=0; i<arr.length;i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
        System.out.println("Enter the value you want to search in the array");


        int val = sc.nextInt();
        int first = 0;
        int last = arr.length - 1;
        int count = 0;
        int mid = 0;
        while(first <= last) {
            mid = (first + last)/2;
            if(arr[mid] == val) {
                System.out.println("WE FOUND THE VALUE "+val+" IN THE GIVEN
ARRAY");

                count++;
                break;
            }
            if(arr[mid] > val) {
                last = mid - 1;
            }
            if(arr[mid] < val) {
                first = mid + 1;
            }
        }
        if(count == 0) {
            System.out.println("THE VALUE "+val + " IS NOT FOUND IN THE ARRAY");
        }
    }
}
```

Question-3:

Write a program to insert an element in the given Array.

Solution:

```
package LabFile;

import java.util.*;
public class ArrayInsertion {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		int pos,val,i,size;
		int [] arr1 = new int[10];
		Scanner sc = new Scanner(System.in);
		System.out.println("THIS IS FOR ARRAY INSERTION\n");
		System.out.println("Enter the Size of Array:");
		size = sc.nextInt();
		System.out.println("Enter Values to Insert in the Array:");
		for(i=0;i<size;i++) {
			arr1[i] = sc.nextInt();
		}

		for(i = 0 ;i<size; i++) {
			System.out.print(arr1[i] + " ");
		}

		System.out.println("\nEnter the value you want to insert");
		val = sc.nextInt();
		System.out.println("Select the position to Insert");
		pos = sc.nextInt();
		if(pos<=size) {
			i = size - 1;
			while (i >= pos) {
				arr1[i+1] = arr1[i];
				i--;
			}

			arr1[pos] = val;
			size = size + 1;

			System.out.println("Insertion is Successfull");
			System.out.println("New Array (after Insertion)");
			for(i = 0 ;i<size; i++) {
				System.out.print(arr1[i] + " ");
			}
		}
		else {
			System.out.println("position not found");
		}
	}
}
```

Question-4:

Write a program to delete an element in the given Array.

Solution:

```
package LabFile;

import java.util.*;
public class Arraydeletion {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                int i, size, pos,val;
                int[] arr1 = new int[10];
                Scanner sc = new Scanner(System.in);
                System.out.println("THIS IS FOR ARRAY DELETION\n");
                System.out.println("Enter the size of the Array:");
                size = sc.nextInt();

                System.out.println("Enter the values to insert in Array");
                for(i=0;i<size;i++) {
                        arr1[i] = sc.nextInt();
                }

                for(i=0;i<size;i++) {
                        System.out.print(arr1[i] + " ");
                }

                System.out.println("\nEnter the position you want to delete from the Array");
                pos = sc.nextInt();

                if (pos <= size-1) {
                        i = pos + 1;
                        while(i <= size) {
                                arr1[i-1] = arr1[i];
                                i++;
                        }
                        size = size - 1;

                        System.out.println("Deletion succesfull");
                        System.out.println("New Array (after deletion)");

                        for(i=0;i<size;i++) {
                                System.out.print(arr1[i] + " ");
                                }
                }
                else {
                        System.out.println("position not found in the array");
                }
        }
}
```

Question-5:

Write a program to merge two arrays into a single Array.

Solution:

```
package LabFile;


import java.util.*;
public class TwoArrayMerging {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		System.out.println("              MERGING OF TWO SINGLE ARRAYS\n");
		System.out.println("Enter the size of First Array");
		int size1 = sc.nextInt();

		int[] arr1 = new int[size1];
		System.out.println("Enter the values to insert in First Array");
		for(int i=0;i<size1;i++) {
			arr1[i] = sc.nextInt();
		}
		System.out.println("Enter the size of Second Array");
		int size2 = sc.nextInt();

		int[] arr2 = new int[size2];
		System.out.println("Enter the values to insert in Second Array");
		for(int i=0;i<size2;i++) {
			arr2[i] = sc.nextInt();
		}
		System.out.println("First Array : ");
		for(int i=0;i<size1;i++) {
			System.out.print(arr1[i] + " ");
		}
		System.out.println();
		System.out.println("Second Array : ");
		for(int i=0;i<size2;i++) {
			System.out.print(arr2[i] + " ");
		}
		int[] arr3 = new int[arr1.length + arr2.length];
		for(int i=0,k=0;i<arr1.length;i++) {
			arr3[k] = arr1[i];
			k++;
		}
		for(int j=0,k=arr1.length;k<(arr1.length + arr2.length);k++) {
			arr3[k] = arr2[j];
			j++;
		}

		System.out.println();
		System.out.println("New Array (after merging)");
		for(int i=0;i<(arr1.length + arr2.length);i++) {
			System.out.print(arr3[i] + " ");
		}
	}
}
```

Question-6:

Write a program to merge two sorted Arrays into one sorted Array.

Solution:

```
package LabFile;
import java.util.*;
public class SortedArrayMerging {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the size of First Array");
		int size1 = sc.nextInt();
		System.out.println("Enter the size of Second Array");
		int size2 = sc.nextInt();
		int[] arr1 = new int[size1];
		int[] arr2 = new int[size2];
		System.out.println("Enter the values for First Array");
		for(int i=0;i<size1;i++) {
			arr1[i] = sc.nextInt();
		}
		System.out.println("Enter the values for Second Array");
		for(int i=0;i<size2;i++) {
			arr2[i] = sc.nextInt();
		}
		System.out.println("First Array : ");
		for(int i=0;i<size1;i++) {
			System.out.print(arr1[i] + " ");
		}
		System.out.println();
		System.out.println("Second Array : ");
		for(int i=0;i<size2;i++) {
			System.out.print(arr2[i] + " ");
		}
		int[] arr3 = new int[arr1.length + arr2.length];
		int i=0,j=0,k=0;
		while(i < arr1.length && j < arr2.length) {
			if(arr1[i] < arr2[j]) {
				arr3[k] = arr1[i];
				k++;
				i++;
			}
			else {
				arr3[k] = arr2[j];
				k++;
				j++;
			}
		}

		while(i < arr1.length) {
			arr3[k] = arr1[i];
			k++;
			i++;
		}
		while(j < arr2.length) {
			arr3[k] = arr2[j];
			k++;
			j++;
```

```
                }
                System.out.println();
                System.out.println("Sorted Merged Arrary");
                for(int l=0;l<arr3.length;l++) {
                        System.out.print(arr3[l] + " ");
                }
        }

}
```

Question-7:

Write a program to search an element in the Array using Iterative and Recursive Binary Search.

Solution:

```
package LabFile;

import java.util.*;
public class IRBinarySearch {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Scanner sc = new Scanner(System.in);
                System.out.println("        Iterative and Recursive Binary Search\n");
                System.out.println("Enter the Size of Array:");
                int size = sc.nextInt();
                int[] arr = new int[size];
                System.out.println("Enter the values of Array (Ascending Order): ");
                for(int i=0; i<arr.length;i++) {
                        arr[i] = sc.nextInt();
                }
                System.out.println();
                System.out.println("The Array : ");
                for(int i=0; i<arr.length;i++) {
                        System.out.print(arr[i] + " ");
                }
                System.out.println();
                System.out.println("Enter the value you want to search in the array:");
                int val = sc.nextInt();
                int first = 0;
                int last = arr.length - 1;
                int count = 0;
                int mid = 0;
                while(first <= last) {
                        mid = (first + last)/2;
                        if(arr[mid] == val) {
                                System.out.println("We found the search value : "+val+ " in the Array");
                                count++;
                                break;
                        }
                        if(arr[mid] > val) {
                                last = mid - 1;
                        }
                        if(arr[mid] < val) {
                                first = mid + 1;
                        }
                }
                if(count == 0) {
                        System.out.println("The value "+val+ " is not found in the Array");
                }
```

```
    }

}
```

Question-8:

Write a menu driven program to implement QUEUE using Arrays that performs following operations (a) INSERT (b) DELETE (c) TRAVERSAL (d) PEEP (e) ISFULL (f) ISEMPTY.

Solution:

```
package LabFile;


import java.util.*;

public class QueueMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.println("                    Queue Menu\n");
        System.out.println("Enter the size of Queue");
        int size = sc.nextInt();
        int[] que = new int[size];
        char ch;
        int front=-1,rear=-1,val=0,count=0;
        do {
            System.out.println("Select any Queue Operations using array");
            System.out.println("1. enque");
            System.out.println("2. deque");
            System.out.println("3. peek");
            System.out.println("4. check empty");
            System.out.println("5. check full");
            System.out.println("6. total no. of elements in the queue");
            System.out.println("7. traversal");
            int choice = sc.nextInt();
            switch(choice) {
                case 1 :

                    System.out.println("Enter the Integer Value for adding in Queue");
                    val = sc.nextInt();
                    if(rear == size-1) {
                        System.out.println("Queue is Overflow.");
                    }
                    else if(front== -1 && rear==-1) {
                        front = 0;
                        rear = 0;
                        que[rear] = val;
                        count++;
                    }
                    else {
                        rear = rear + 1;
                        que[rear] = val;
                        count++;
                    }

                    System.out.println("Queue : ");
                    for(int i=front;i<=rear;i++) {
                        System.out.print(que[i] + " ");
                    }
                    break;


                case 2 :
                    if(front == -1 && rear == -1) {
                        System.out.println("Queue is Underflow.");
                    }
                    else if(front == rear) {
                        front = -1;
                        rear = -1;
                        count--;
                    }
                    else {
                        System.out.println("deque (deleted) element = "+
que[front]);

                        front = front + 1;
                        count--;
                    }
                    System.out.println("Queue : ");
                    for(int i=front; i<=rear;i++) {
                        System.out.print(que[i] + " ");
                    }
                    break;


                case 3 :
                    System.out.println("peek element = "+que[front]);
                    break;


                case 4 :
                    System.out.println("empty status = "+(size-(rear+1))+" spaces are
remaining.");

                    break;


                case 5 :
                    if(rear == size-1) {
                        System.out.println("full status = "+true);
                    }
                    else {
                        System.out.println("Queue is not full. \nit still has "+(size-
(rear+1))+" spaces remaining.");

                    }
                    break;


                case 6 :
                    System.out.println("Total no. of element are = "+count);
                    break;


                case 7 :
                    System.out.println("Traversal : ");
                    if(front == -1 && rear == -1) {
                        System.out.println("Queue is Underflow.");
                    }
                    int i = front;
                    System.out.println("Queue : ");
                    while(i <= rear) {
                        System.out.print(que[i]+" ");
                        i = i+1;
                    }
                    break;


                default :
                    System.out.println("Wrong entry");
                    break;
            }
            System.out.println("\n"+"\n"+"do you want to continue (y/n) ");
            ch = sc.next().charAt(0);

        }while(ch == 'Y' || ch == 'y');
    }

}
```

Question-9:

A. **Write a menu driven program to implement Circular Queue using Arrays that performs following operations. (a) INSERT (b) DELETE (c) DISPLAY (d) PEEP (e) ISFULL (f) ISEMPTY.**

B. **Write a menu driven program to implement a program for Stack that performs following operations using Array. (a)PUSH (b) POP (c) PEEP (d) DISPLAY (e) ISFULL (f) ISEMPTY**

Solution-A:

```
package LabFile;


import java.util.*;
class CircularQueueMethod{
    private int f, r, i, count=0, size;
    private int[] cirQue;

    CircularQueueMethod(int n){
        f = -1;
        r = -1;
        size = n;
        cirQue = new int[size];
    }

    public void enque(int num) {
        if(f == 0 && r == size-1) {
            System.out.println("Circular Queue is Overflow.");
        }
        else if(f == -1 && r == -1) {
            f = 0;
            r = 0;
            cirQue[r] = num;
            count++;
        }
        else if(f !=0 && r==size-1) {
            r = 0;
            cirQue[r] = num;
            count++;
        }
        else {
            r = r + 1;
            cirQue[r] = num;
            count++;
        }

    }

    public void deque() {
        if(f == -1 && r == -1) {
            System.out.println("Circular Queue is Underflow.");
        }
        else if(f==r) {
            System.out.println("deque (deleted) item : "+ cirQue[f]);
            f = -1;
            r = -1;
            count--;
        }
        else if(f == size-1) {
            System.out.println("deque (deleted) item : "+ cirQue[f]);
            f = 0;
            count--;
        }
        else {
            System.out.println("deque (deleted) item : "+ cirQue[f]);
            f = f + 1;
            count--;
        }
    }

    public int peek() {
        return cirQue[f];
    }

    public boolean underflow() {
        return r == -1;
    }

    public boolean overflow() {
        return f == r + 1;
    }

    public void traversal() {
        if(f == -1 && r == -1) {
            System.out.print("Circular Queue is Underflow.");
        }
        else if(r > f) {
            i = f;
            while(i <= r) {
                System.out.print(cirQue[i] + " ");
                i = i + 1;
                count++;
            }
        }
        else {
            i = f;
            while(i <= size-1) {
                System.out.print(cirQue[i] + " ");
                i = i + 1;
                count++;
            }
            i = 0;
            while(i <= r) {
                System.out.print(cirQue[i] + " ");
                i = i + 1;
                count++;
            }
        }
    }

    public int totalele() {
        return count;
    }
}

public class CircularQueue_Menu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.println("        Circular Queue Menu\n");
        System.out.println("Enter the size of Circular Queue");
        int size = sc.nextInt();
        char ch;
        int front=0,rear=0,val=0;
        CircularQueueMethod cqm = new CircularQueueMethod(size);
        do {
            System.out.println("Circular Queue Operations using array");
            System.out.println("1. enque");
            System.out.println("2. deque");
            System.out.println("3. peek");
            System.out.println("4. check empty");
            System.out.println("5. check full");
            System.out.println("6. total no. of elements in the queue");
            System.out.println("7. traversal");
            int choice = sc.nextInt();
            switch(choice) {
                case 1 :
                    System.out.println("Enter the Integer Value for adding in Queue");
                    val = sc.nextInt();
                    cqm.enque(val);
                    break;
                case 2 :
                    System.out.println("deque element = ");
                    cqm.deque();
                    break;
                case 3 :
                    System.out.println("peek element = " + cqm.peek());
                    break;
                case 4 :
                    System.out.println("empty status = " + cqm.underflow());
                    break;
                case 5 :
                    System.out.println("full status = " + cqm.overflow());
                    break;
                case 6 :
                    System.out.println("Total no. of element are = " + cqm.totalele());
                    break;
                case 7 :
                    System.out.println("Traversal : ");
                    cqm.traversal();
                    break;
                default :
                    System.out.println("Wrong entry");
                    break;
            }

            System.out.println("\n\ndo you want to continue (y/n) ");
            ch = sc.next().charAt(0);

        }while(ch == 'Y' || ch == 'y');
    }

}
```

Solution-B:

```
package LabFile;
import java.util.*;
class StackMethod{
    private int top, size,count,i;
    private int[] stack;
    StackMethod(int ak){
        size = ak;
        top = -1;
        stack = new int[ak];
    }
    public void push(int num) {
        if(top == size - 1) {
            System.out.println("Stack is Overflow.");
        }
        else {
            top = top + 1;
            stack[top] = num;
        }
    }
    public void pop() {
        if(top == -1) {
            System.out.println("Stack is Underflow.");
        }
        else {
            System.out.println("pop item : "+ stack[top]);
            top = top - 1;
        }
    }
    public int peek() {
        return stack[top];
    }

    public boolean overflow() {
        return top == size-1;
    }

    public boolean underflow() {
        return top == -1;
    }

    public int totalEle() {
        if(top == -1) {
            System.out.println("Stack is Underflow.");
            return 0;
        }
        else {
            for(int j = top; j>=0 ;j--) {
                count++;
            }
            return count;
        }
    }
    public void traversal() {
        if(top == -1) {
            System.out.println("Stack is Underflow.");
        }
        else {
            i=top;
            while(i>=0) {
                System.out.print(stack[i] + " ");
                i = i - 1;
            }
        }
    }
    public void display() {
        System.out.println("\nStack : ");
        for(int i=top;i>=0;i--) {
            System.out.print(stack[i] + " ");
        }
    }
}
public class StackMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.println("Stack Menu");
        System.out.println("Enter the size of Stack");
        int sizestk = sc.nextInt();
        StackMethod stk = new StackMethod(sizestk);
        char ch;
        int val=0;
        do {
            System.out.println("Stack Operations using array");
            System.out.println("1. push");
            System.out.println("2. pop");
            System.out.println("3. peek");
            System.out.println("4. check empty");
            System.out.println("5. check full");
            System.out.println("6. total no. of elements in the queue");
            System.out.println("7. traversal");
            int choice = sc.nextInt();
            switch(choice) {
                case 1 :
                    System.out.println("Enter the Integer Value for adding in Stack");
                    val = sc.nextInt();
                    stk.push(val);
                    break;
                case 2 :
                    System.out.println("pop (deleted) element = ");
                    stk.pop();
                    break;
                case 3 :
                    System.out.println("peek element = "+stk.peek());
                    break;
                case 4 :
                    System.out.println("empty status = " + stk.underflow());
                    break;
                case 5 :
                    System.out.println("full status = " + stk.overflow());
                    break;
                case 6 :
                    System.out.println("Total no. of element are = "+stk.totalEle());
                    break;
                case 7 :
                    System.out.print("Traversal : ");
                    stk.traversal();
                    break;
                default :
                    System.out.println("Wrong entry");
                    break;
            }
            stk.display();
            System.out.println("\ndo you want to continue (y/n) ");
            ch = sc.next().charAt(0);


        }while(ch == 'Y' || ch == 'y');
    }
}
```

Question-10:

Write a program to convert infix notation to postfix notation using Stack.

Solution:

```
package LabFile;
import java.util.Scanner;
class stackl
{
    private char[] a;
    private int top,m;
    private int count= 0;
    public stackl(int max)

    {
        m = max;
        a = new char[m];
        top = -1;
    }

    public void push(char t)
    {
        a[++top] = t;
        count ++;
    }

    public char pop()
    {
        count-- ;
        return(a[top--]);

    }
```

```
              public char peek()
              {
                            return(a[top]);
              }

              public boolean isEmpty()
              {
                            return (top == -1);
              }
              public char Size()
              {
                            return (char) count;
              }
}
public class InfixToPostStack {

    public static boolean isOperator(char c) {
        return c == '+' || c == '-' || c == '*' || c == '/' || c == '^' ||c == '$' || c == '(' || c == ')';
    }

    private static int getPrecedence(char ch) {
        switch (ch) {
        case '-':
        case '+':
            return 1;

        case '/':
        case '*':
            return 2;

        case '$':
        case'^':
            return 3;
        }
        return -1;
    }
    public static  String reverse(String inflix)
    {
    String original=inflix;
      String reverse = "";


        int length = original.length();

        for (int i = length - 1 ; i >= 0 ; i--)
          reverse = reverse + original.charAt(i);
                            return reverse;


    }

    private static boolean isOperand(char ch) {
        return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9');
    }

    public static String convertToPostfix(String infix) {
              stackl sc = new stackl(infix.length());
        StringBuffer postfix = new StringBuffer(infix.length());
        char c;
        for (int i = 0; i < infix.length(); i++) {
            c = infix.charAt(i);

            if (isOperand(c))
            {
                postfix.append(c);
            }
            else if (c == '('){
                sc.push(c);
                postfix.append('(');
                        }

            else if (c == ')') {

                while (!sc.isEmpty() && sc.peek() != '(') {
                    postfix.append(sc.pop());

                }

                if (!sc.isEmpty() && sc.peek() != '(')
                    return null;
                else if(!sc.isEmpty())
                    sc.pop();
                postfix.append(')');
            }

            else if (isOperator(c))
            {
                if (!sc.isEmpty() && getPrecedence(c) < getPrecedence(sc.peek())) {
                    postfix.append(sc.pop());
                }
                sc.push(c);
            }
        }

        while (!sc.isEmpty()) {
            postfix.append(sc.pop());
        }
        return postfix.toString();
    }

public static void main(String[] args) {
              String s;

        Scanner inp=new Scanner(System.in);
        System.out.println("Enter the infix expression ");
         s=inp.nextLine();

        System.out.println("Postfix expression:- "+convertToPostfix(s));

    }}
```

Question-11:

Write a program to convert infix notation to prefix notation using Stack.

Solution:

```
package LabFile;


import java.util.Scanner;
class stacklk
{
            private char[] a;
            private int top,m;
            private int count= 0;
            public stacklk(int max)

            {
                        m = max;
                        a = new char[m];
                        top = -1;
            }

            public void push(char t)
            {
                        a[++top] = t;
                        count ++;
            }

            public char pop()
            {
                        count-- ;
                        return(a[top--]);

            }

            public char peek()
            {
                        return(a[top]);
            }

            public boolean isEmpty()
            {
                        return (top == -1);
            }
            public char Size()
            {
                        return (char) count;
            }
}
public class InfixToPreStack {

   public static boolean isOperator(char c) {
      return c == '+' || c == '-' || c == '*' || c == '/' || c == '^' ||c == '$' || c == '(' || c == ')';
   }

   private static int getPrecedence(char ch) {
      switch (ch) {
      case '-':
      case '+':
         return 1;
```

```
    case '/':
    case '*':
        return 2;

    case '$':
    case'^':
        return 3;
    }
    return -1;
}
public static  String reverse(String inflix)
{
String original=inflix;
  String reverse = "";


    int length = original.length();

    for (int i = length - 1 ; i >= 0 ; i--)
      reverse = reverse + original.charAt(i);
                        return reverse;


}

private static boolean isOperand(char ch) {
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9');
}

 public static String convertToPrefix(String infix) {
          stacklk sc = new stacklk(infix.length());
    StringBuffer prefix = new StringBuffer(infix.length());
    reverse(infix);
    char c;

    for (int i = 0; i < infix.length(); i++) {
        c = infix.charAt(i);

        if (isOperand(c))
        {
            prefix.append(c);
        }
        else if (c == '('){
            sc.push(c);
            prefix.append('(');
                    }

        else if (c == ')') {

            while (!sc.isEmpty() && sc.peek() != '(') {
                prefix.append(sc.pop());

            }

            if (!sc.isEmpty() && sc.peek() != '(')
                return null;
            else if(!sc.isEmpty())
                sc.pop();
            prefix.append(')');
        }

        else if (isOperator(c))
        {
            if (!sc.isEmpty() && getPrecedence(sc.peek())>= getPrecedence(c)  ) {
                prefix.append(sc.pop());
            }
            sc.push(c);
        }
    }

    while (!sc.isEmpty()) {
        prefix.append(sc.pop());
    }
    String r = prefix.toString();
    reverse(r);
    return r;
}



public static void main(String[] args) {

    Scanner inp=new Scanner(System.in);

     System.out.println("Enter the infix expression ");
           String s=inp.nextLine();
           System.out.println("Prefix expression:- "+convertToPrefix(s));

}
}
```

Question-12:

Write a program to evaluate given postfix notation using Stack.

Solution:

```
package LabFile;


import java.util.Stack;

public class PostfixStack_12
{
    // Method to evaluate value of a postfix expression
    static int evaluatePostfix(String exp)
    {
        //create a stack
        Stack<Integer> stack=new Stack<>();

        // Scan all characters one by one
        for(int i=0;i<exp.length();i++)
        {
            char c=exp.charAt(i);

            // If the scanned character is an operand (number here),
            // push it to the stack.
            if(Character.isDigit(c))
            stack.push(c - '0');
```

```
      //  If the scanned character is an operator, pop two
      // elements from stack apply the operator
      else
      {
        int val1 = stack.pop();
        int val2 = stack.pop();

        switch(c)
        {
          case '+':
          stack.push(val2+val1);
          break;

          case '-':
          stack.push(val2- val1);
          break;

          case '/':
          stack.push(val2/val1);
          break;

          case '*':
          stack.push(val2*val1);
          break;
       }
      }
    }
    return stack.pop();
  }

  // Driver program to test above functions
  public static void main(String[] args)
  {
    String exp="231*+9-";
    System.out.println("postfix evaluation: "+evaluatePostfix(exp));
  }
}
```

Question-13:

Write a menu driven program to implement following operations on the singly Linked List.

a) Insert a node at the front of the Linked List.

b) Insert a node at the end of the Linked List.

c) Insert a node such that Linked List is in ascending order. (according to info. Field)

d) Delete a first node of the Linked List.

e) Delete a node before specified position.

f) Delete a node after specified position.

g) Traversal of Linked List

Solution:

```
package LabFile;


import java.util.*;
```

```
class Node{
    protected int data;
    public Node link;

    public Node() {
        data = 0;
        link = null;
    }
    public Node(int d, Node n) {
        data = d;
        link = n;
    }
    public void setdata(int d) {
        data = d;
    }
    public int getdata() {
        return data;
    }
    public void setlink(Node n) {
        link = n;
    }
    public Node getlink() {
        return link;
    }

}

class Linked_list{
    public Node head;
    public int size;

    Linked_list(){
        head = null;
        size = 0;
    }

    public void insertAtStart(int a) {
        Node new_node = new Node(a,null);
        new_node.setlink(head);
        head = new_node;
        size++;
    }

    public void insertAtLast(int b) {
        Node new_node1 = new Node(b,null);
        Node ptr = head;
        while(ptr.getlink() != null) {
            ptr = ptr.getlink();
        }
        ptr.setlink(new_node1);
        new_node1.setlink(null);
        size++;
    }

    public void insertAfterGivenNode(int c,int sval) {
        Node new_node2 = new Node(c,null);
        Node ptr = head;
        while(ptr.getdata() != sval) {
            ptr = ptr.getlink();
        }
        new_node2.setdata(c);
        new_node2.setlink(ptr.getlink());
        ptr.setlink(new_node2);
        size++;
    }

    public void insertBeforeGivenNode(int d, int sval) {
        Node new_node3 = new Node(d,null);
        Node ptr = head;
        while(ptr.getlink().getdata() != sval) {
            ptr = ptr.getlink();
        }
        new_node3.setdata(d);
        new_node3.setlink(ptr.getlink());
        ptr.setlink(new_node3);
        size++;
    }

    public void delAtFirst() {
        Node ptr = head;
        head = head.getlink();
        size--;
    }

    public void delAtLast() {
        Node ptr = head;
        while(ptr.getlink().getlink() != null) {
            ptr = ptr.getlink();
        }
        ptr.setlink(null);
        size--;
    }

    public void delAfterNode(int sv) {
        Node ptr = head;
        while(ptr.getdata() != sv) {
            ptr = ptr.getlink();
        }
        Node temp = ptr.getlink();
        ptr.setlink(ptr.getlink().getlink());
        size--;
    }

    public void delBeforeNode(int sv1) {
        Node ptr = head, pptr = head, ppptr = head;
        while(ptr.getdata() != sv1) {
            ppptr = pptr;
            pptr = ptr;
            ptr = ptr.getlink();
        }
        ppptr.setlink(ptr);
        size--;
    }

    public void display() {
        Node ptr = head;
        //System.out.print("->");
        while(ptr.getlink() != null) {
            System.out.print(ptr.getdata() + " -> ");
            ptr = ptr.getlink();
        }
        System.out.print(ptr.getdata() + "\n");
    }
}
public class singlyLinkedListMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        Linked_list linlist = new Linked_list();
        char ch;
        do {
            System.out.println("    singly Linked List Menu\n");
            System.out.println("Select an option to proceed:");
            System.out.println("        1. Insertion in LinkedList.");
            System.out.println("        2. Deletion in LinkedList.");
            System.out.println("        3. Total no. of Elements in the LinkedList.");
            int choice = sc.nextInt();
            switch (choice) {
                case 1 :
                    System.out.println("\nSelect the option below for Insertion : \n");
                    do {
                        System.out.println("1. Insertion at First Position : ");
                        System.out.println("2. Insertion at Last Position : ");
                        System.out.println("3. Insertion After a given Node : ");
                        System.out.println("4. Insertion Before a give Node : ");
                        int choice1 = sc.nextInt();
                        switch (choice1) {
                            case 1 :
                                System.out.println("Insert Value to add
at First.");

                                int val = sc.nextInt();
                                linlist.insertAtStart(val);
                                linlist.display();
                                break;

                            case 2 :
                                System.out.println("Insert Value to add
at Last.");

                                int val1 = sc.nextInt();
                                linlist.insertAtLast(val1);
                                linlist.display();
                                break;

                            case 3 :
                                System.out.println("Insert Value to add
after the given Node.");

                                int val2 = sc.nextInt();
                                System.out.println("Insert Search

Value.");

                                int val3 = sc.nextInt();
                                linlist.insertAfterGivenNode(val2,

val3);

                                linlist.display();
                                break;

                            case 4 :
                                System.out.println("Insert Value to add
before the given Node.");
```

```
                                int val4 = sc.nextInt();
                                System.out.println("Insert the Search
Value.");

                                int val5 = sc.nextInt();
                                linlist.insertBeforeGivenNode(val4,
val5);

                                linlist.display();
                                break;
                        }
                        System.out.println("Do you want to continue the insertion
loop (y/n) ");

                        ch = sc.next().charAt(0);
                    }while(ch == 'Y' || ch == 'y');
                    break;


                case 2 : {
                    System.out.println("\nSelect the option below for Deletion : ");
                    do {
                        System.out.println("1. Deletion at First Position : ");
                        System.out.println("2. Deletion at Last Position : ");
                        System.out.println("3. Deletion After a Given Position :
");

                        System.out.println("4. Deletion Before a Given Position :
");

                        int choice2 = sc.nextInt();
                        switch (choice2) {
                            case 1 :
                                System.out.println("Item deleted at First
Position : ");

                                linlist.delAtFirst();
                                linlist.display();
                                break;
                            case 2 :
                                System.out.println("Item deleted at Last
Position : ");

                                linlist.delAtLast();
                                linlist.display();
                                break;
                            case 3 :
                                System.out.println("Enter the Position :
" );

                                int sv = sc.nextInt();
                                System.out.println("Item deleted after
the Given Position : ");

                                linlist.delAfterNode(sv);
                                linlist.display();
                                break;
                            case 4 :
                                System.out.println("Enter the Position :
");

                                int sv1 = sc.nextInt();
                                System.out.println("Item deleted before
the Given Position : ");

                                linlist.delBeforeNode(sv1);
                                linlist.display();
                                break;
                        }
                        System.out.println("Do you want to continue deletion
loop (y/n) ");
```

```
                        ch = sc.next().charAt(0);
                    }while(ch == 'Y' || ch == 'y');
                    break;
                    }
                case 3 : {
                    System.out.println("\nTotal Elements : "+ linlist.size);
                    break;
                }
            }
            System.out.println();
            linlist.display();
            System.out.println("Do you want to continue the menu (y/n) ");
            ch = sc.next().charAt(0);
        }while(ch == 'Y' || ch == 'y');
    }

}
```

Question-14:

Write a menu driven program to implement Stack using Linked List.

Solution:

```
package LabFile;


import java.util.*;

class NodeS{
    protected int data;
    public NodeS link;

    public NodeS() {
        data = 0;
        link = null;
    }
    public NodeS(int d, NodeS n) {
        data = d;
        link = n;
    }
    public void setdata(int d) {
        data = d;
    }
    public int getdata() {
        return data;
    }
    public void setlink(NodeS n) {
        link = n;
    }
    public NodeS getlink() {
        return link;
    }

}

class StackLinked_list{
    protected NodeS top;
    public int size,max;
    StackLinked_list(int n){
        top = null;
        size = 0;
        max = n;
    }

    public boolean isEmpty() {
        return top == null;
    }

    public boolean isFull() {
        return size == max;
    }

    public int getSize() {
        return size;
    }

    public void insertAtStart(int a) {
        if(isFull()) {
            System.out.println("Stack is Overflow.");
        }
        else {
            NodeS new_node = new NodeS(a,null);
            if(top == null) {
                top = new_node;
            }
            else {
                top.setlink(new_node);
                top = top.getlink();
                size++;
            }
        }
}

    public int delAtFirst() {
        if(isEmpty()) {
            System.out.println("Stack is Underflow.");
            return 0;
        }
        else {
            NodeS ptr = top;
            top = top.getlink();
            size--;
            return ptr.getdata();
        }
    }


    public void display() {
        if(top == null) {
            System.out.println("Stack is Underflow.");
        }
        else {
            NodeS ptr = top;
            while(ptr != null) {
                System.out.print(+ptr.getdata()+" ");
                ptr = ptr.getlink();
            }
        }
    }

}

public class StackLinkedListMenu {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of Stack.");
        int size = sc.nextInt();
        StackLinked_list Stl = new StackLinked_list(size);
        char ch;
        do {
            System.out.println("Select an option to proceed");
            System.out.println("1. Insertion in Stack.");
            System.out.println("2. Deletion in Stack.");
            System.out.println("3. Total no. of Elements.");
            int choice = sc.nextInt();
            switch (choice) {
                case 1 :
                    System.out.println("\nInsert Value to add at First.");
                    int valS = sc.nextInt();
                    Stl.insertAtStart(valS);
                    break;

                case 2 : {
                    System.out.println("\nItem deleted at First Position : ");
                    Stl.delAtFirst();
                    break;
                    }
                case 3 : {
                    System.out.println("\nTotal Elements : "+Stl.getSize());
                    break;
                }
            }
            System.out.println();
            Stl.display();
            System.out.println("\nDo you want to continue the menu (y/n) ");
            ch = sc.next().charAt(0);
        }while(ch == 'Y' || ch == 'y');
    }
}
```

Question-15:

Write a menu driven program to implement Queue using Linked List.

Solution:

```
package LabFile;

import java.util.*;

class NodeQ{
        protected int data;
        public NodeQ link;

        public NodeQ() {
                data = 0;
                link = null;
        }
        public NodeQ(int d, NodeQ n) {
                data = d;
                link = n;
        }
        public void setdata(int d) {
                data = d;
        }
        public int getdata() {
                return data;
        }
        public void setlink(NodeQ n) {
                link = n;
        }
        public NodeQ getlink() {
                return link;
        }
}

class QueueLinked_list{
        protected NodeQ front, rear;
        public int size,max;

        QueueLinked_list(int n){
                front = null;
                rear = null;
                size = 0;
                max = n;
        }

        public boolean isEmpty() {
                return front == null;
        }

        public boolean isFull() {
                return size == max;
        }

        public int getSize() {
                return size;
        }

        public void insertAtLast(int a) {
```

```
        if(isFull()) {
            System.out.println("Queue is Overflow.");
        }
        else {
            NodeQ new_node = new NodeQ(a,null);
            if(rear == null) {
                rear = new_node;
                front = new_node;
                size++;
            }
            else {
                rear.setlink(new_node);
                rear = rear.getlink();
                size++;
            }
        }
    }

    public int delAtFirst() {
        if(isEmpty()) {
            System.out.println("Queue is Underflow.");
            return -1;
        }
        else {
            NodeQ ptr = front;
            front = front.getlink();
            if(front == null) {
                rear = null;
                size--;
            }
            size--;
            return ptr.getdata();
        }
    }

    public int peek() {
        if(isEmpty()) {
            System.out.println("Queue is Underflow.");
            return 0;
        }
        else {
            return front.getdata();
        }
    }


    public void display() {
        System.out.println("Stack with LinkedList : ");
        if(size == 0) {
            System.out.println("Queue is Empty.");
        }
        else {
            NodeQ Sptr = front;
            while(Sptr != rear.getlink()) {
                System.out.print(Sptr.getdata() + " ");
                Sptr = Sptr.getlink();
            }
            System.out.println();
        }
    }
```

```
}

public class QueueLinkedList {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.println("Queue Linked List Menu\n");
        System.out.println("Enter the size of Queue.");
        int size = sc.nextInt();
        QueueLinked_list qlt = new QueueLinked_list(size);
        char ch;
        do {
            System.out.println("Select an option to proceed\n");
            System.out.println("1. Insertion in Queue.");
            System.out.println("2. Deletion in Queue.");
            System.out.println("3. Peek.");
            System.out.println("4. Total no. of Elements.");
            int choice = sc.nextInt();
            switch (choice) {
                case 1 :
                    System.out.println("\nInsert Value to add at First.");
                    int valS = sc.nextInt();
                    qlt.insertAtLast(valS);
                    break;

                case 2 : {
                    System.out.println("\nItem deleted at First Position : ");
                    qlt.delAtFirst();
                    break;
                    }
                case 3 : {
                    System.out.println("\nThe First Element is : "+ qlt.peek());

                    break;
                }
                case 4 : {
                    System.out.println("\nThe Total number of elements are : "+
qlt.getSize());

                    break;
                }
            }
            System.out.println();
            qlt.display();
            System.out.println("\nDo you want to continue the menu (y/n) ");
            ch = sc.next().charAt(0);
        }while(ch == 'Y' || ch == 'y');
    }

}
```

Question-16:

Write a program to implement following operations on the doubly Linked List.

a) Insert a node at the front of the Linked List.

b) Insert a node at the end of the Linked List.

c) Delete a last node of the Linked List.

d) Delete a node before specified position.

e) Traversal of Linked List

Solution:

```
package LabFile;

import java.util.Scanner;
class DouNode {
		protected int data;
		public DouNode prev,next;

		DouNode(){
			data = 0;
			prev = null;
			next = null;
		}

		DouNode(int d, DouNode n1, DouNode n2){
			data = d;
			prev = n1;
			next = n2;
		}

		public void setdata(int d) {
			data = d;
		}

		public int getdata() {
			return data;
		}

		public void setprev(DouNode n1) {
			prev = n1;
		}

		public DouNode getprev() {
			return prev;
		}

		public void setnext(DouNode n2) {
			next = n2;
		}

		public DouNode getnext() {
			return next;
		}
}

class DoubleLinkedList {
```

```java
public DouNode head;
public int size;

DoubleLinkedList(){
        head = null;
        size = 0;
}

public void insertAtFirstDoubLinList(int a) {
        DouNode new_node = new DouNode(a,null,null);
        new_node.setnext(head);
        new_node.setprev(new_node);
        head = new_node;
        size++;
}

public void insertAtLastDoubLinList(int b) {
        DouNode new_node1 = new DouNode(b,null,null);
        DouNode ptr = head;
        while(ptr.getnext() != null) {
                ptr = ptr.getnext();
        }
        ptr.setnext(new_node1);
        new_node1.setprev(ptr);
        size++;
}

public void insertAtBeforeDoubLinList(int c, int sval) {
        DouNode new_node2 = new DouNode(c,null,null);
        DouNode ptr = head;
        while(ptr.getdata() != sval) {
                ptr = ptr.getnext();
        }
        new_node2.setprev(ptr.getprev());
        new_node2.setnext(ptr);
        ptr.getprev().setnext(new_node2);
        ptr.setprev(new_node2);
        size++;
}

public void insertAfterDoubLinList(int d, int sv) {
        DouNode new_node3 = new DouNode(d,null,null);
        DouNode ptr = head;
        while(ptr.getdata() != sv) {
                ptr = ptr.getnext();
        }
        new_node3.setprev(ptr);
        new_node3.setnext(ptr.getnext());
        ptr.getnext().setprev(new_node3);
        ptr.setnext(new_node3);
        size++;
}

public void deleteFirst() {
        DouNode temp = head;
        temp.getnext().setprev(null);
        head = temp.getnext();
        size--;
}
```

```java
    public void deleteLast() {
        DouNode ptr = head;
        while(ptr.getnext() != null) {
            ptr = ptr.getnext();
        }
        ptr.getprev().setnext(null);
        size--;
    }

    public void deletebefore(int sv1) {
        DouNode ptr = head;
        while(ptr.getdata() != sv1) {
            ptr = ptr.getnext();
        }
        ptr.setprev(ptr.getprev().getprev());
        ptr.getprev().getprev().setnext(ptr);
        size--;
    }

    public void deleteafter(int sv) {
        DouNode ptr = head;
        while(ptr.getdata() != sv) {
            ptr = ptr.getnext();
        }
        DouNode temp = ptr.getnext();
        ptr.setnext(ptr.getnext().getnext());
        ptr.getnext().setprev(ptr);
        size--;
    }

    public void display() {
        DouNode ptr = head;
        while(ptr.getnext() != null) {
            System.out.print(ptr.getdata() + " -> ");
            ptr = ptr.getnext();
        }
        System.out.println(ptr.getdata() + "\n");
    }

}


public class DoubleLinkedListMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        DoubleLinkedList dll = new DoubleLinkedList();
        char c;
        do {
            System.out.println("Doubly Linked List Menu Choose the Option : ");
            System.out.println("1. Insertion to Doubly LinkedList : ");
            System.out.println("2. Deletion to Doubly LinkedList : ");
            System.out.println("3. Total Number of Elements : ");
            int choice = sc.nextInt();
            switch (choice) {
                case 1 : {
                    do {
                        System.out.println("1. Insertion at Fist Position.");
                        System.out.println("2. Insertion at Last Position.");
                        System.out.println("3. Insertion before a given Position.");
                        System.out.println("4. Insertion after a given Position.");
                        int choice1 = sc.nextInt();
                        switch (choice1) {
                            case 1 : {
                                System.out.println("Enter the value to
add in Linked List.");

                                int v = sc.nextInt();
                                dll.insertAtFirstDoubLinList(v);
                                dll.display();
                                break;
                            }

                            case 2 : {
                                System.out.println("Enter the value to
add at the last in Linked List.");

                                int v1 = sc.nextInt();
                                dll.insertAtLastDoubLinList(v1);
                                dll.display();
                                break;
                            }

                            case 3 : {
                                System.out.println("Enter the value of
position.");

                                int sval = sc.nextInt();
                                System.out.println("Enter the value to
add before the Position.");

                                int v2 = sc.nextInt();
                                dll.insertAtBeforeDoubLinList(v2,
sval);

                                dll.display();
                                break;
                            }

                            case 4 : {
                                System.out.println("Enter the value of
position.");

                                int sv = sc.nextInt();
                                System.out.println("Enter the value to
add after the Position.");

                                int v3 = sc.nextInt();
                                dll.insertAfterDoubLinList(v3, sv);
                                dll.display();
                                break;
                            }
                        }
                        System.out.println("do you want to continue insertion
menu (y/n) : ");

                        c = sc.next().charAt(0);
                    }while(c == 'Y' || c == 'y');
                    break;
                }
                case 2 : {
                    do {
                        System.out.println("1. Deletion at First Position.");
                        System.out.println("2. Deletion at Last Position. ");
                        System.out.println("3. Deletion Before a Given
Position.");

                        System.out.println("4. Deletion After a Given Position.");
                        int choice2 = sc.nextInt();
                        switch (choice2) {
                            case 1 : {
                                System.out.println("Item Deleted at
First Position.");

                                dll.deleteFirst();
                                dll.display();
                                break;
                            }

                            case 2 : {
                                System.out.println("Item Deleted at Last
Position.");

                                dll.deleteLast();
                                dll.display();
                                break;
                            }

                            case 3 : {
                                System.out.println("Enter the
Position.");

                                int p = sc.nextInt();
                                dll.deletebefore(p);
                                System.out.println("Item Deleted at

before the Given Position : ");

                                dll.display();
                                break;
                            }

                            case 4 : {
                                System.out.println("Enter the
Position.");

                                int p1 = sc.nextInt();
                                dll.deleteafter(p1);
                                System.out.println("Item Deleted after

the Given Position : ");

                                dll.display();
                                break;
                            }
                        }

                        System.out.println("do you want to continue deletion

menu (y/n) : ");

                        c = sc.next().charAt(0);
                    }while(c == 'Y' || c == 'y');
                    break;
                }

                case 3 : {
                    System.out.println("Total Number of Elements are : " + dll.size);
                    break;
                }
            }
            dll.display();
            System.out.println("do you want to continue Doubly LinkedList Menu (y/n) : ");
            c = sc.next().charAt(0);
        }while(c == 'Y' || c == 'y');
    }

}
```

Question-17:

Write a program to implement following operations on the circular Linked List.

a) Insert a node at the end of the Linked List.

b) Insert a node before specified position.

c) Delete a first node of the Linked List.

d) Delete a node after specified position.

e) Traversal of Linked List

Solution:

```
package LabFile;


import java.util.*;

class NodeC{
	protected int data;
	public NodeC link;

	public NodeC() {
		data = 0;
		link = null;
	}
	public NodeC(int d) {
		data = d;
		//link = n;
	}
	public void setdata(int d) {
		data = d;
	}
	public int getdata() {
		return data;
	}
	public void setlink(NodeC n) {
		link = n;
	}
	public NodeC getlink() {
		return link;
	}
}

class CircularLinked_list{
	public NodeC head;
	public int size;

	CircularLinked_list(){
		//head = null;
		size = 0;
	}

	public void cirInsertFirst(int a) {
		NodeC new_nodeCir = new NodeC(a);
		if(head == null) {
			head = new_nodeCir;
```

```
            new_nodeCir.setlink(head);
            return ;
        }

        new_nodeCir.setlink(head);
        NodeC ptrC = head;
        while(ptrC.getlink() != head) {
            ptrC = ptrC.getlink();
        }
        head = new_nodeCir;
        ptrC.setlink(head);
        size++;
    }

    public void cirInsertLast(int b) {
        NodeC new_node1 = new NodeC(b);

        if(head == null) {
            head = new_node1;
            return ;
        }

        NodeC ptr = head;
        while(ptr.getlink() != head) {
            ptr = ptr.getlink();
        }
        ptr.setlink(new_node1);
        new_node1.setlink(head);
    }

    public void insertAfterGivenNode(int c,int sval) {
        NodeC new_node2 = new NodeC(c);

        if(head == null) {
            head = new_node2;
            return ;
        }

        NodeC ptr = head;
        while(ptr.getdata() != sval) {
            ptr = ptr.getlink();
        }
        new_node2.setdata(c);
        new_node2.setlink(ptr.getlink());
        ptr.setlink(new_node2);
        size++;
    }

    public void insertBeforeGivenNode(int d, int sval) {
        NodeC new_node3 = new NodeC(d);

        if(head == null) {
            head = new_node3;
            return ;
        }

        NodeC ptr = head;
        while(ptr.getlink().getdata() != sval) {
            ptr = ptr.getlink();
        }
        new_node3.setdata(d);
        new_node3.setlink(ptr.getlink());
        ptr.setlink(new_node3);
        size++;
    }

    public void cirDeleteFirst() {
        NodeC ptr = head;
        while(ptr.getlink() != head) {
            ptr = ptr.getlink();
        }
        ptr.setlink(head.getlink());
        head = head.getlink();
    }

    public void cirDeleteLast() {
        NodeC ptr = head,pptr=head;
        while(ptr.getlink() != head) {
            pptr = ptr;
            ptr = ptr.getlink();
        }
        pptr.setlink(head);
    }

    public void Cirdisplay() {
        NodeC ptr = head;
        //System.out.print("->");
        while(ptr.getlink() != head) {
            System.out.print(ptr.getdata() + " -> ");
            ptr = ptr.getlink();
        }
        System.out.print(ptr.getdata() + "\n");
    }
}

public class CircularLinkedListMenu {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        CircularLinked_list Cirlinlist = new CircularLinked_list();
        char ch;
        do {
            System.out.println("Circular LinkedList Menu.");
            System.out.println("1. Insertion in LinkedList.");
            System.out.println("2. Deletion in LinkedList.");
            int choice = sc.nextInt();
            switch (choice) {
                case 1 :
                    System.out.println("\nSelect the option below for Insertion : ");
                    do {
                        System.out.println("\n1. Insertion at First Position : ");
                        System.out.println("2. Insertion at Last Position : ");
                        System.out.println("3. Insertion After a given Node : ");
                        System.out.println("4. Insertion Before a give Node : ");
                        int choice1 = sc.nextInt();
                        switch (choice1) {
                            case 1 :
                                System.out.println("Insert Value to add
at First.");

                                int val = sc.nextInt();
                                Cirlinlist.cirInsertFirst(val);
                                Cirlinlist.Cirdisplay();
                                break;

                            case 2 :
                                System.out.println("Insert Value to add
at Last.");

                                int val1 = sc.nextInt();
                                Cirlinlist.cirInsertLast(val1);
                                Cirlinlist.Cirdisplay();
                                break;

                            case 3 :
                                System.out.println("Insert Value to add
after the given Node.");

                                int val2 = sc.nextInt();
                                System.out.println("Insert Search

Value.");

                                int val3 = sc.nextInt();
                                Cirlinlist.insertAfterGivenNode(val2,

val3);

                                Cirlinlist.Cirdisplay();
                                break;

                            case 4 :
                                System.out.println("Insert Value to add
before the given Node.");

                                int val4 = sc.nextInt();
                                System.out.println("Insert the Search

Value.");

                                int val5 = sc.nextInt();
                                Cirlinlist.insertBeforeGivenNode(val4,

val5);

                                Cirlinlist.Cirdisplay();
                                break;
                        }
                        System.out.println("Do you want to continue the insertion
loop (y/n) ");

                        ch = sc.next().charAt(0);
                    }while(ch == 'Y' || ch == 'y');
                    break;


                case 2 : {
                    System.out.println("\nSelect the option below for Deletion : ");
                    do {
                        System.out.println("1. Deletion at First Position : ");
                        System.out.println("2. Deletion at Last Position : ");
                        int choice2 = sc.nextInt();
                        switch (choice2) {
                            case 1 :
                                System.out.println("Item deleted at First
Position : ");

                                Cirlinlist.cirDeleteFirst();
                                Cirlinlist.Cirdisplay();
                                break;
                            case 2 :
                                System.out.println("Item deleted at Last
Position : ");

                                Cirlinlist.cirDeleteLast();
                                Cirlinlist.Cirdisplay();
                                break;
                        }
                        System.out.println("Do you want to continue deletion
loop (y/n) ");

                        ch = sc.next().charAt(0);
                    }while(ch == 'Y' || ch == 'y');
                    break;
                }
            }
            System.out.println();
            Cirlinlist.Cirdisplay();
            System.out.println("Do you want to continue the menu (y/n) ");
            ch = sc.next().charAt(0);
        }while(ch == 'Y' || ch == 'y');
    }
}
```

Question-18:

Write a program which create Binary Tree.

Solution:

```
package LabFile;

import java.util.*;
class BT{
    public BT left,right;
    int data;

    public BT(int n) {
        left = null;
        right = null;
        data = n;
    }

    public void setleft(BT l) {
        left = l;
    }

    public BT getleft() {
        return left;
    }

    public void setright(BT r) {
        right = r;
    }

    public BT getright() {
        return right;
    }

    public void setdata(int d) {
        data = d;
    }

    public int getdata() {
        return data;
    }
```

```
	public void inorder(BT r) {
		if(r != null) {
			inorder(r.getleft());
			System.out.print(r.getdata()+" ");
			inorder(r.getright());
		}
	}

}
public class BinaryTree {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		BT bt = staticBT();
		System.out.print("Inorder : ");
		bt.inorder(bt);

	}
	public static BT staticBT() {
		BT root = new BT(15);
		BT nodeB = new BT(25);
		BT nodeC = new BT(17);
		BT nodeD = new BT(20);
		BT nodeE = new BT(30);
		BT nodeF = new BT(45);
		BT nodeG = new BT(37);
		root.setleft(nodeB);
		root.setright(nodeC);
		nodeB.setleft(nodeD);
		nodeB.setright(nodeE);
		nodeC.setright(nodeF);
		nodeF.setleft(nodeG);
		return root;

	}

}
```

Question-19:

Write a program to implement recursive and non-recursive Binary Tree traversing methods in-order, pre-order and post-order traversal.

Solution:

```
package LabFile;


import java.util.*;

class NodeBT {
	 protected int data;
	protected NodeBT left, right;

	public NodeBT() {
		data = 0;
		left = null;
		right = null;
	}
```

```
    public NodeBT(int n) {
        data = n;
        left = null;
        right = null;
    }

    void setdata(int n1) {
        data = n1;
    }

    int getdata() {
        return data;
    }

    void setleft(NodeBT l) {
        left = l;
    }

    NodeBT getleft() {
        return left;
    }

    void setright(NodeBT r) {
        right = r;
    }

    NodeBT getright() {
        return right;
    }
}

class BinaryTr {
    int c = 0;
    NodeBT root = null;

    public void insert(int a) {
        c = c+1;
        root = insert(root,a);
    }

    public NodeBT insert(NodeBT n, int v1) {
        if(n == null) {
            n = new NodeBT(v1);
        }
        else if(n.left == null) {
            n.left = insert(n.left,v1);
        }
        else if(n.right == null) {
            n.right = insert(n.right,v1);
        }
        else
            n.left = insert(n.left,v1);
        return n;
    }

    int sum = 0;
    public void inorder(NodeBT r1) {
        if(r1 != null) {
            inorder(r1.getleft());
            System.out.print(r1.getdata()+" ");
            inorder(r1.getright());
        }
    }

    public void preorder(NodeBT r2) {
        if(r2 != null) {
            System.out.print(r2.getdata() + " ");
            preorder(r2.getleft());
            preorder(r2.getright());
        }
    }

    public void postorder(NodeBT r3) {
        if(r3 != null) {
            postorder(r3.getleft());
            postorder(r3.getright());
            System.out.print(r3.getdata() + " ");
        }
    }

}
public class BinaryTreeMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        char ch;
        BinaryTr BT = new BinaryTr();

        do {
            System.out.println("1. Insertion to Binary Tree.");
            System.out.println("2. Inorder of Binary Tree.");
            System.out.println("3. Preorder of Binary Tree.");
            System.out.println("4. Postorder of Binary Tree.");


            int choice = sc.nextInt();
            switch(choice) {
            case 1 :
                System.out.println("Enter the value to insert.");
                int v1 = sc.nextInt();
                BT.insert(v1);
                break;

            case 2 :
               BT.inorder(BT.root);
               break;

            case 3 :
                BT.preorder(BT.root);
                break;

            case 4 :
                BT.postorder(BT.root);
                break;

            default:
                System.out.println("Invalid Input");
                break;
            }
```

```
                        System.out.print("\nInorder : ");
                        BT.inorder(BT.root);
                        System.out.println("\nDo you want to perform any operations (y/n) : ");
                        ch = sc.next().charAt(0);

                }while(ch == 'y' || ch == 'Y');
        }

}
```

Question-20:

Write a menu driven program to implement Binary Search Tree and its Traversal.

Solution:

```
package LabFile;


import java.util.*;

class NodeBT {
         protected int data;
        protected NodeBT left, right;

        public NodeBT() {
                data = 0;
                left = null;
                right = null;
        }

        public NodeBT(int n) {
                data = n;
                left = null;
                right = null;
        }

        void setdata(int n1) {
                data = n1;
        }

        int getdata() {
                return data;
        }

        void setleft(NodeBT l) {
                left = l;
        }

        NodeBT getleft() {
                return left;
        }

        void setright(NodeBT r) {
                right = r;
        }

        NodeBT getright() {
                return right;
        }
```

```
}

class BinaryTr {
    int c = 0;
    NodeBT root = null;

    public void insert(int a) {
        c = c+1;
        root = insert(root,a);
    }

    public NodeBT insert(NodeBT n, int v1) {
        if(n == null) {
            n = new NodeBT(v1);
        }
        else if(n.left == null) {
            n.left = insert(n.left,v1);
        }
        else if(n.right == null) {
            n.right = insert(n.right,v1);
        }
        else
            n.left = insert(n.left,v1);
        return n;
    }

    int sum = 0;
    public void inorder(NodeBT r1) {
        if(r1 != null) {
            inorder(r1.getleft());
            System.out.print(r1.getdata()+" ");
            inorder(r1.getright());
        }
    }

    public void preorder(NodeBT r2) {
        if(r2 != null) {
            System.out.print(r2.getdata() + " ");
            preorder(r2.getleft());
            preorder(r2.getright());
        }
    }

    public void postorder(NodeBT r3) {
        if(r3 != null) {
            postorder(r3.getleft());
            postorder(r3.getright());
            System.out.print(r3.getdata() + " ");
        }
    }

}
public class BinaryTreeMenu {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        char ch;
        BinaryTr BT = new BinaryTr();
        do {
            System.out.println("1. Insertion to Binary Tree.");
            System.out.println("2. Inorder of Binary Tree.");
            System.out.println("3. Preorder of Binary Tree.");
            System.out.println("4. Postorder of Binary Tree.");


            int choice = sc.nextInt();
            switch(choice) {
            case 1 :
                System.out.println("Enter the value to insert.");
                int v1 = sc.nextInt();
                BT.insert(v1);
                break;

            case 2 :
              BT.inorder(BT.root);
              break;

            case 3 :
                BT.preorder(BT.root);
                break;

            case 4 :
                BT.postorder(BT.root);
                break;

            default:
                System.out.println("Invalid Input");
                break;
            }
            System.out.print("\nInorder : ");
            BT.inorder(BT.root);
            System.out.println("\nDo you want to perform any operations (y/n) : ");
            ch = sc.next().charAt(0);

        }while(ch == 'y' || ch == 'Y');
    }

}
```

Question-21:

Write a menu driven program to implement AVL Tree and its Traversal.