

MAR THOMA INSTITUTE OF INFORMATION TECHNOLOGY
CHADAYAMANGALAM P.O, AYUR
(Affiliated to The University of Kerala & Approved by the AICTE)



DEPARTMENT OF COMPUTER APPLICATIONS

Bonafide record of work done by

*(Reg. No.....) in the.....Lab of
Mar Thoma Institute of Information Technology, Ayur during the
year.....*

Staff in charge

Head of the Department

*Submitted for the MCA Degree Practical Examination held at Mar Thoma Institute of
Information Technology, Ayur on.....*

Internal Examiner

External Examiner

INDEX

SI.NO	CONTENTS	PAGE.NO	DATE	REMARKS
1	PERFORM ARRAY OPERATIONS			
2	STACK USING ARRAY			
3	QUEUE USING ARRAY			
4	CIRCULAR QUEUE			
5	DEQUEUE (Double Ended queue)			
6	EVALUATE POSTFIX EXPRESSION			
7	REVERSAL OF STRING			
8	SELECTION SORT			
9	INSERTION SORT			
10	BUBBLE SORT			
11	LINEAR SEARCH			
12	BINARY SEARCH			
13	QUICK SORT			
14	MERGE SORT			
15	SINGLY LINKED LIST			
16	DOUBLY LINKED LIST			
17	CIRCULAR LINKED LIST			
18	REVERSE A SINGLY LINKED LIST			
19	STACK USING LINKED LIST			
20	QUEUE USING LINKED LIST			
21	BINARY SEARCH TREE			
22	CONVERT INFIX TO POSTFIX			
23	BINARY SEARCH TREE TRAVERSAL			

Program:

```
import java.util.Scanner;

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

        System.out.println("Enter the array limit");
        int n = sc.nextInt();
        int a[] = new int[n+1];

        System.out.println("Enter Array Elements");
        for(int i = 0; i<n; i++)
        {
            a[i]=sc.nextInt();
        }

        System.out.print("Array Elements Are - " + " ");
        for(int i=0; i<n; i++) {
            System.out.print(a[i] + " ");
        }
        System.out.println(" ");

        System.out.println("Enter the index for insertion");
        int index = sc.nextInt();

        System.out.println("Enter the element for insertion");
        int element = sc.nextInt();
        for(int i=n-1; i>=index-1; i--) {
            a[i+1] = a[i];
        }
        a[index - 1] = element;
        System.out.print("Array elements after insertion - ");
        for(int i=0; i<=n; i++) {
            System.out.print(a[i] + " ");
        }
        System.out.println("");

        System.out.println("Enter the index for deletion");
        int del = sc.nextInt();
        for(int i=del-1; i<=n-1; i++) {
            a[i] = a[i+1];
        }
        System.out.print("");

        System.out.print("Array elements after deletion - ");
        for(int i=0; i<n; i++) {
            System.out.print(a[i] + " ");
```

```
}  
}  
}
```

OUTPUT:

```
Enter the array limit  
6  
Enter Array Elements  
45  
55  
65  
75  
85  
95  
Array Elements Are - 45 55 65 75 85 95  
Enter the index for insertion  
3  
Enter the element for insertion  
70  
Array elements after insertion - 45 55 70 65 75 85 95  
Enter the index for deletion  
4  
Array elements after deletion - 45 55 70 75 85 95
```

Program:

```
class Stack
{
    static final int MAX=100;
    int top;
    int stack[]=new int[MAX];

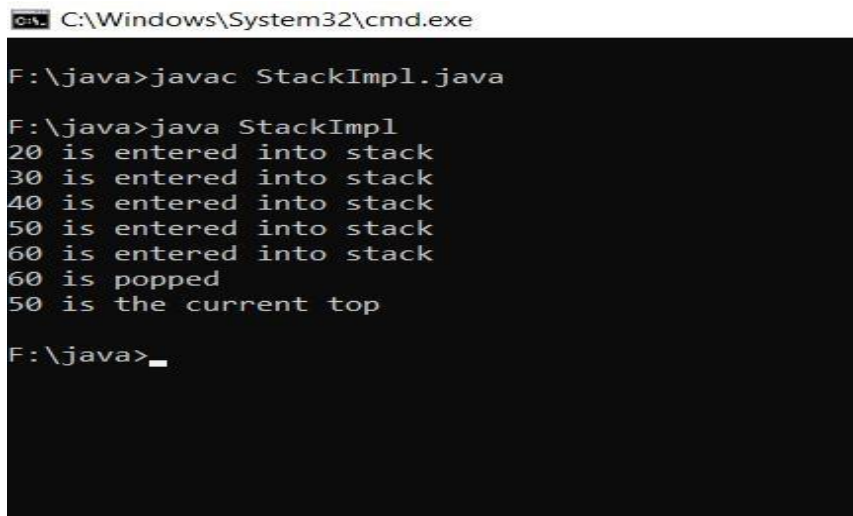
    Stack()
    {
        top=-1;
    }
    public boolean isEmpty()
    {
        return top<0 ;
    }
    public boolean isFull()
    {
        return top==MAX-1;
    }
    int push(int x)
    {
        if (isFull())
        {
            System.out.println("stack is overflow");
            return 0;
        }
        else
        {
            System.out.println(x + " is entered into stack");
            return stack[++top]=x;
        }
    }
    int pop()
    {
        if (isEmpty())
        {
            System.out.println("stack is underflow");
            return 0;
        }
        else
        {
            int x =stack[top--];
            return x ;
        }
    }
}
```

```

    }
    int peek()
    {
        if(isEmpty())
        {
            System.out.println("stack is underflow");
            return 0;
        }
        else
        {
            int x=stack[top];
            return x;
        }
    }
}
class StackImpl
{
    public static void main(String[] args)
    {
        Stack s = new Stack();
        s.push(20);
        s.push(30);
        s.push(40);
        s.push(50);
        s.push(60);
        System.out.println(s.pop() + " is popped");
        System.out.println(s.peek() + " is the current top");
    }
}

```

OUTPUT:



```

C:\Windows\System32\cmd.exe
F:\java>javac StackImpl.java
F:\java>java StackImpl
20 is entered into stack
30 is entered into stack
40 is entered into stack
50 is entered into stack
60 is entered into stack
60 is popped
50 is the current top
F:\java>_

```

Program:

```
class Queue
{
    static final int MAX=10;

    int front,rear;
    int queue[]=new int[MAX];

    Queue()
    {
        front=-1;
        rear=-1;
    }
    public boolean isEmpty()
    {
        return front<0 && rear<0 ;
    }
    public boolean isFull()
    {
        return rear==MAX-1;
    }
    int Enqueue(int x)
    {
        if (isFull())
        {
            System.out.println("Queue is overflow");
            return 0;
        }
        else
        {
            front=0;
```

```

        System.out.println(x + " is entered into Queue");
        return queue[++rear]=x;
    }
}

int Dequeue()
{
    if (isEmpty())
    {
        System.out.println("Queue is underflow");
        return 0;
    }
    else
    {
        return queue[front++] ;
    }
}

int peek()
{
    if(isEmpty())
    {
        System.out.println("Queue is underflow");
        return 0;
    }
    else
    {
        int x=queue[front];
        return x;
    }
}

void display()

```



```

    {
        if(isEmpty())
        {
            System.out.println("Queue is underflow");
        }
        else
        {
            System.out.print("current queue elements: ");
            for (int i=front;i<=rear;i++)
            {
                System.out.print(queue[i]+" ");
            }
        }
    }
}

```

```

class QueueImpl
{
    public static void main(String[] args)
    {
        Queue s = new Queue();
        s.Enqueue(100);
        s.Enqueue(200);
        s.Enqueue(300);
        s.Enqueue(400);
        s.Enqueue(500);
        System.out.println(s.Dequeue() + " is popped");
        System.out.println(s.Dequeue() + " is popped");
        System.out.println(s.peek() + " is the current front");
        s.display();
    }
}

```

```
    }  
}
```

OUTPUT:

```
100 is entered into Queue  
200 is entered into Queue  
300 is entered into Queue  
400 is entered into Queue  
500 is entered into Queue  
100 is popped  
200 is popped  
300 is the current front  
current queue elements: 300 400 500
```

Program:

```
class CircularQueue{

    int size,front,rear;

    int q_arr[];

    CircularQueue(int size){
        this.size = size;
        front = -1;
        rear = -1;
        this.q_arr = new int[size];
    }

    boolean isFull(){
        if(front ==0 && rear == size -1){
            return true;
        }
        if(front == rear + 1){
            return true;
        }
        return false;
    }

    boolean isEmpty(){
        if(front == -1){
            return true;
        }
        else{
            return false;
        }
    }
}
```

```

void enqueue(int data){
    if(isFull()){
        System.out.println("Queue is full");
    }
    else{
        if(front == -1){
            front = 0;
        }
        rear = (rear + 1)%size;
        q_arr[rear] = data;
        System.out.println("\n"+data + " inserted");
    }
}

int dequeue(){
    if(isEmpty()){
        System.out.println("Queue is empty");
        return -1;
    }
    else{
        int data = q_arr[front];
        if(front == rear){
            front = -1;
            rear = -1;
        }
        else{
            front = (front+1)%size;
        }
        return data;
    }
}

```

```

void display(){
    int i;
    if(isEmpty()){
        System.out.println("Queue is empty");
    }
    else{
        System.out.print("The Queue is: ");
        for(i = front; i != rear; i = (i+1)%size){
            System.out.print(q_arr[i]+ " ");
        }
        System.out.print(q_arr[i]);
    }
}

```

```

public static void main(String args[]){
    CircularQueue q = new CircularQueue(5);

    q.dequeue();

    q.enqueue(8);
    q.enqueue(4);
    q.enqueue(2);
    q.enqueue(9);
    q.enqueue(1);
    q.enqueue(10);
    q.display();

    q.dequeue();
    q.dequeue();
}

```

```
        System.out.println();
        q.display();
        q.enqueue(10);
        q.enqueue(11);
        q.enqueue(11);
        q.display();
        q.dequeue();
        System.out.println();
        q.display();
        q.enqueue(12);
        q.display();
    }
}
```

Output:

```
Queue is empty
8 inserted
4 inserted
2 inserted
9 inserted
1 inserted
Queue is full
The Queue is: 8 4 2 9 1
The Queue is: 2 9 1
10 inserted
11 inserted
Queue is full
The Queue is: 2 9 1 10 11
The Queue is: 9 1 10 11
12 inserted
The Queue is: 9 1 10 11 12
```

Program:

```
class Deque
{
    static final int MAX = 100;

    int arr[];
    int front;
    int rear;
    int size;

    public Deque(int size)
    {
        arr = new int[MAX];
        front = -1;
        rear = 0;
        this.size = size;
    }

    boolean isFull()
    {
        return ((front == 0 && rear == size-1)||
                front == rear+1);
    }

    boolean isEmpty ()
    {
        return (front == -1);
    }

    void insertfront(int key)
```

```

{
    if (isFull())
    {
        System.out.println("Overflow");
        return;
    }

    if (front == -1)
    {
        front = 0;
        rear = 0;
    }

    else if (front == 0)
        front = size - 1 ;

    else
        front = front-1;

    arr[front] = key ;
}

void insertrear(int key)
{
    if (isFull())
    {
        System.out.println(" Overflow ");
        return;
    }

```



```
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }

    else if (rear == size-1)
        rear = 0;

    else
        rear = rear+1;

    arr[rear] = key ;
}

void deletefront()
{
    if (isEmpty())
    {
        System.out.println("Queue Underflow");
        return ;
    }

    if (front == rear)
    {
        front = -1;
        rear = -1;
    }

    else
```

```

        if (front == size - 1)
            front = 0;

        else
            front = front + 1;
    }

void deleterear()
{
    if (isEmpty())
    {
        System.out.println(" Underflow");
        return ;
    }

    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else if (rear == 0)
        rear = size - 1;
    else
        rear = rear - 1;
}

int getFront()
{
    if (isEmpty())
    {

```

```
        System.out.println(" Underflow");
        return -1 ;
    }
    return arr[front];
}
```

```
int getRear()
{
    if(isEmpty() || rear < 0)
    {
        System.out.println(" Underflow");
        return -1 ;
    }
    return arr[rear];
}
```

```
public static void main(String[] args)
{
```

```
    Deque dq = new Deque(5);
```

```
    System.out.println("Insert element at rear end : 5 ");
    dq.insertrear(5);
```

```
    System.out.println("insert element at rear end : 10 ");
    dq.insertrear(10);
```

```
    System.out.println("get rear element : "+ dq.getRear());
```

```
    dq.deleterear();
```

```
System.out.println("After delete rear element new rear is : " + dq.getRear());

System.out.println("inserting element at front end");
dq.insertfront(15);

System.out.println("get front element: " +dq.getFront());

dq.deletefront();
System.out.println("After delete front element new front is : " + dq.getFront());

}
}
```

Output:

```
Insert element at rear end : 5
insert element at rear end : 10
get rear element : 10
After delete rear element new rear is : 5
inserting element at front end
get front element: 15
After delete front element new front is : 5
```

Program:

```
import java.util.*;

class StackPost
{
    static final int MAX=100; int top;
    int stack[] =new int[MAX];
    StackPost()
    {
        top=-1;
    }
    public boolean isEmpty()
    {
        return top<0 ;
    }
    public boolean isFull()
    {
        return top==MAX-1;
    }

    int Postfix(String exp)
    { for(int i=0;i<exp.length();i++)
        {
            char c=exp.charAt(i);

            if(Character.isDigit(c))
            {
                int a = c - '0';
                push(a);
            }
            else
            {
```

```

        int v1=pop(); int
        v2=pop(); switch(c)
        {
            case '+': push(v2+v1);
            break;

            case '-': push(v2-v1);
            break;

            case '*': push(v2*v1);
            break;

            case '/': push(v2/v1);
            break;
        }
    }

    return pop();
}

int push(int x)
{ if (isFull())
    {
        System.out.println("stack is overflow"); return 0;
    }
    else
    {
        return stack[++top]=x;
    }
}

int pop()
{
    if (isEmpty())
    {

```

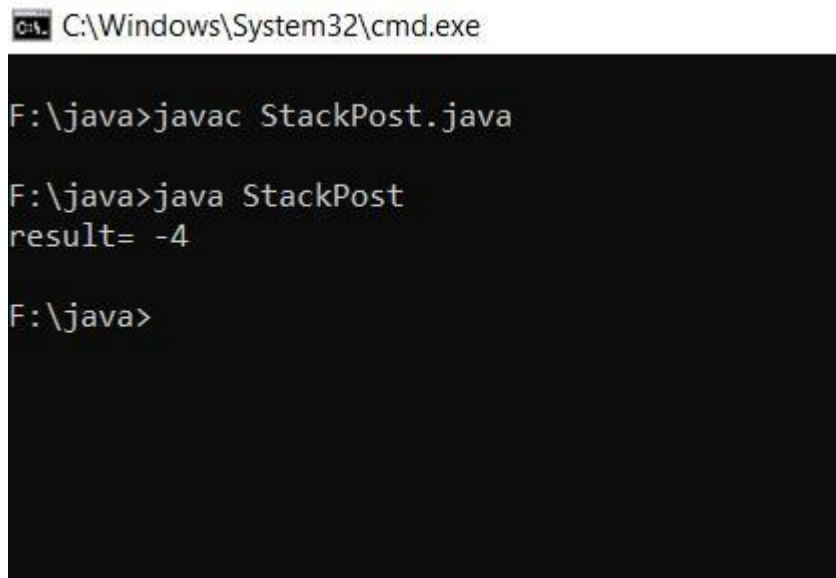
```

        System.out.println("stack is underflow"); return 0;
    }
    else
    {
        int x =stack[top--];
        return x ;
    }
}

public static void main(String[] args)
{
    StackPost s= new StackPost(); int
    post=s.Postfix("231*+9-");
    System.out.println("result= "+post);
}
}

```

OUTPUT:



C:\Windows\System32\cmd.exe

```

F:\java>javac StackPost.java

F:\java>java StackPost
result= -4

F:\java>

```

Program:

```
import java.util.*;

class StringRev{

    static String reversal(String str){

        String rev = "";

        int len = str.length();

        for(int i=len -1; i>=0; i--){

            rev = rev + str.charAt(i);

        }

        return rev;

    }

    public static void main(String args[]){

        String str;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the string: ");

        str = sc.nextLine();

        System.out.println("Reversed string: "+reversal(str));

    }

}
```

Output:

```
Enter the string:
hello
Reversed string: olleh
```


Program:

```
import java.util.*;

class SelectionSort
{
    public static void main(String[] args) {
        int temp,i,j; int min;

        Scanner s=new Scanner(System.in);
        System.out.println("Enter the array limit");

        int n =s.nextInt(); int arr[]=new int[n];
        System.out.println("Enter the array");
        for(i=0;i<n;i++)
        { arr[i]=s.nextInt();
        }
        for(i=0;i<arr.length;i++)
        {
            min=i;

            for(j=i+1;j<arr.length;j++)
            { if (arr[j]<arr[min])
                {
                    min=j;
                }
            }

            temp=arr[i];
            arr[i]=arr[min];
            arr[min]=temp;
        }
        System.out.println();
        System.out.println("Sorted Array:");
        for(i=0;i<arr.length;i++) {

            System.out.println(arr[i]);
        }
    }
}
```

}

OUTPUT:

C:\Windows\System32\cmd.exe

```
F:\java>javac SelectionSort.java
```

```
F:\java>java SelectionSort
```

```
Enter the array limit
```

```
6
```

```
Enter the array
```

```
90
```

```
85
```

```
12
```

```
56
```

```
45
```

```
62
```

```
Sorted Array:
```

```
12
```

```
45
```

```
56
```

```
62
```

```
85
```

```
90
```

```
F:\java>
```

Program:

```
import java.util.*;

class InsertionSort
{
    public static void main(String[] args)

    { int i,j,temp;

        Scanner s=new Scanner(System.in);
        System.out.println("Enter the array limit"); int n
        =s.nextInt(); int arr[]=new int[n];

        System.out.println("Enter the array");
        for(i=0;i<n;i++)
        { arr[i]=s.nextInt();
        }

        for ( i = 0; i < arr.length; i++)
        {
            j = i;

            while (j > 0 && arr[j] < arr[j-1])
            {
                temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp; j--;
            }
        }

        System.out.println();

        System.out.println("Sorted Array:");
        for(i=0;i<arr.length;i++)
        {
            System.out.println(arr[i]);
        }
    }
}
```

}

OUTPUT:

```
C:\Windows\System32\cmd.exe

F:\java>javac InsertionSort.java

F:\java>java InsertionSort
Enter the array limit
6
Enter the array
100
56
45
23
34
67

Sorted Array:
23
34
45
56
67
100

F:\java>
```

Program:

```
import java.util.*;

class BubbleSort
{
    public static void main(String[] args) {
        int i,j,temp,m;

        Scanner s=new Scanner(System.in);
        System.out.println("Enter the array limit");

        int n =s.nextInt(); int arr[]=new int[n];
        System.out.println("Enter the array");
        for(i=0;i<n;i++)
        { arr[i]=s.nextInt();
        }

        for(i=0;i<arr.length;i++)
        { for(j=1;j<arr.length-i;j++)
            { if(arr[j-1]>arr[j])
                {
                    temp=arr[j]; arr[j]=arr[j-1];
                    arr[j-1]=temp;
                }
            }
        }

        System.out.println();
        System.out.println("Sorted Array:");
        for(i=0;i<arr.length;i++)
        {
            System.out.println(arr[i]);
        }
    }
}
```

OUTPUT:

C:\Windows\System32\cmd.exe

```
F:\java>javac BubbleSort.java
```

```
F:\java>java BubbleSort
```

```
Enter the array limit
```

```
6
```

```
Enter the array
```

```
45
```

```
21
```

```
35
```

```
12
```

```
89
```

```
78
```

```
Sorted Array:
```

```
12
```

```
21
```

```
35
```

```
45
```

```
78
```

```
89
```

```
F:\java>
```

Program:

```
import java.util.Scanner;

class LinearSearch
{
    public static void main(String[] args)
    {
        int flag=0,i;

        Scanner s=new Scanner(System.in);
        System.out.println("Enter the array limit");

        int n =s.nextInt(); int arr[]=new int[n];
        System.out.println("Enter the array");
        for(i=0;i<n;i++)
        { arr[i]=s.nextInt();
        }

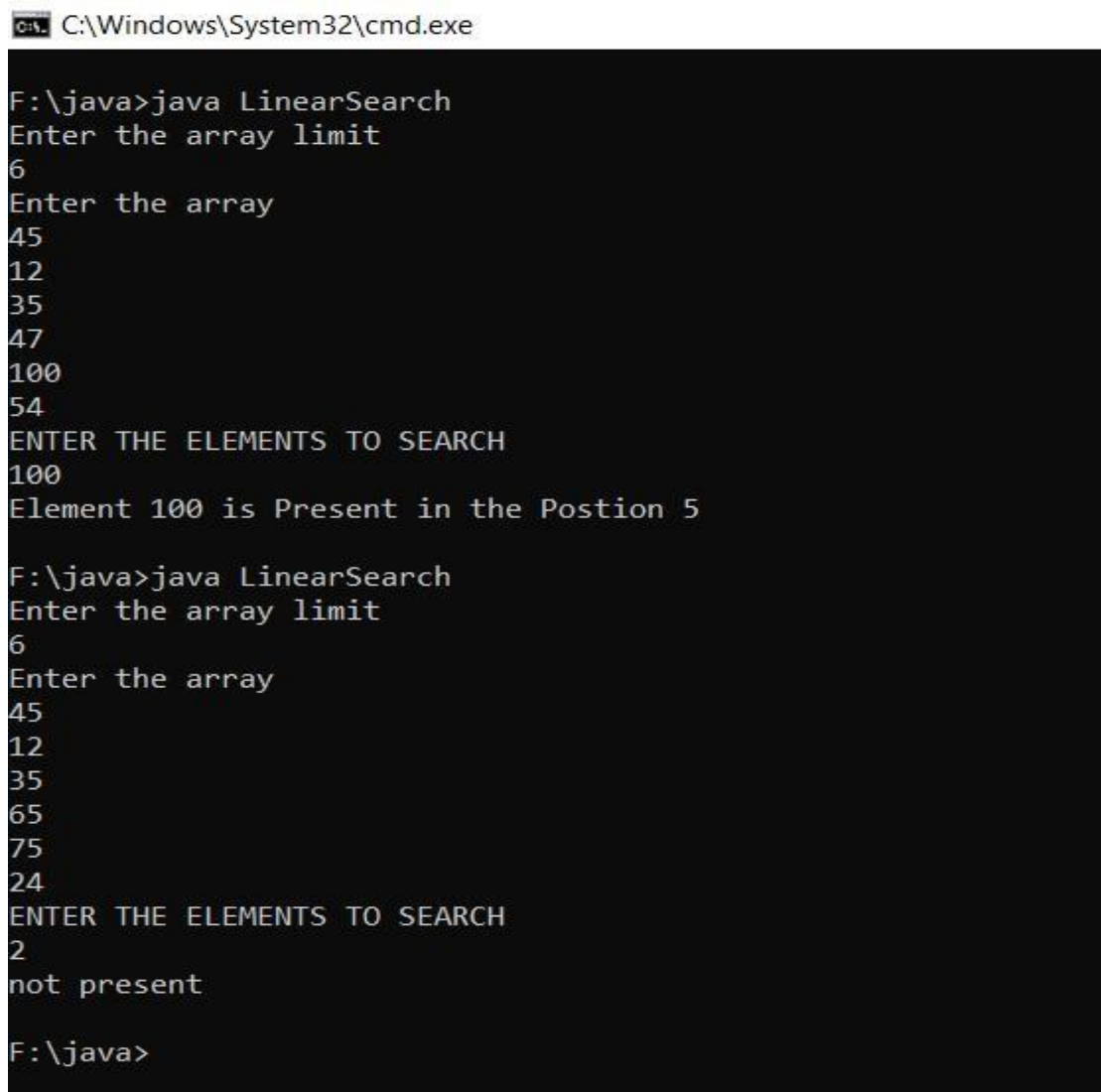
        System.out.println("ENTER THE ELEMENTS TO SEARCH");
        int x=s.nextInt(); for(i=0;i<n;i++)
        {
            if(arr[i]==x)
            {
                flag=i+1;
                break;
            }
            else
            {
                flag=0;
            }
        }
        if(flag==0)
        {
            System.out.println("not present");
        }
    }
}
```

```

    }
else
{
    System.out.println("Element " + x + " is Present in the Postion " +
        flag );
}
}
}
}

```

OUTPUT:



```

C:\Windows\System32\cmd.exe

F:\java>java LinearSearch
Enter the array limit
6
Enter the array
45
12
35
47
100
54
ENTER THE ELEMENTS TO SEARCH
100
Element 100 is Present in the Postion 5

F:\java>java LinearSearch
Enter the array limit
6
Enter the array
45
12
35
65
75
24
ENTER THE ELEMENTS TO SEARCH
2
not present

F:\java>

```


Program:

```
import java.util.*; public class BinarySearch {

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        System.out.println("Enter the size of an array");

        int n = s.nextInt(); int[] arr = new int[n];

        System.out.println("Enter the values in sorted order");

        for (int i = 0; i < n; i++)
        { arr[i] = s.nextInt();
        }

        System.out.println("Enter value to be searched");

        int num = s.nextInt();

        int low = 0; int high = n - 1;
        int mid = 0;

        while (low <= high) {

            mid = (low + high) / 2;
            if (arr[mid] == num) {
                System.out.print(" Value found at " + (mid+1)); break;
            }
            else if (arr[mid] > num) {
                high = mid - 1;

            } else if (num > arr[mid]) {
                low = mid + 1;
            }
        }
    }
}
```

```
        if (low > high) {  
            System.out.println(" Value is not present in the array ");  
        }  
    }  
}
```

OUTPUT:

```
Enter the size of an array  
6  
Enter the values in sorted order  
40  
45  
55  
60  
75  
90  
Enter value to be searched  
75  
Value found at the index 5
```

```
Enter the size of an array  
6  
Enter the values in sorted order  
40  
56  
75  
95  
100  
110  
Enter value to be searched  
200  
Value is not found in the array
```

Program:

```
import java.util.*;

class QuickSort
{
    public static void main(String[] args)
    {
        Scanner S =new Scanner(System.in); System.out.println("enter
        the limit of array");

        int n=S.nextInt(); int
        a[]=new int[n];

        System.out.println("enter the " +n+ " elements");
        for(int i=0;i<=n-1;i++)
        { a[i]=S.nextInt();
        }

        int length=n;

        QuickSort q=new QuickSort();

        q.Quick(a,0,length-1);

        q.diplay(a,length-1);

    }

    int partiton(int a[],int low,int high)
    {
        int pivot=(a[low]);

        while (low<=high)
        {
            while (a[low]<pivot)
            {
                low++;
            }

            while(a[high]>pivot)
            { high--;
```

```

        }
        if (low<=high)
        {
            int temp=a[low];
            a[low]=a[high];
            a[high]=temp;

            low++;

            high--;

        }
    }

    return low;
}

void Quick(int a[],int low,int high)
{
    int p=partiton(a,low,high);
    if(low<p-1)
    {
        Quick(a,low,p-1);

    } if(high>p)
    {
        Quick(a,p,high);
    }
}


void diplay(int a[],int length)
{
    System.out.println("Sorted array:");
    for(int i=0; i<=length;i++)
    {

        System.out.println(a[i]);
    }
}

```

```
    }  
}
```

OUTPUT:

 C:\Windows\System32\cmd.exe

```
F:\java>javac QuickSort.java
```

```
F:\java>java QuickSort
```

```
enter the limit of array
```

```
6
```

```
enter the 6 elements
```

```
20
```

```
30
```

```
10
```

```
70
```

```
50
```

```
90
```

```
Sorted array:
```

```
10
```

```
20
```

```
30
```

```
50
```

```
70
```

```
90
```

```
F:\java>
```

Program:

```
import java.util.*;

class MergeSort
{
    void Sort(int a[],int l,int r)
    { if(l<r)
        {
            int mid=(l+r)/2;
            Sort(a,l,mid);
            Sort(a,mid+1,r);

            Merge(a,l,mid,r);
        }
    }

    void Merge(int a[],int l,int mid,int r)
    {
        int n1=mid-l+1;
        int n2=r-mid;
        int L[]=new int[n1];
        int R[]=new int[n2];
        for(int i=0;i<n1;i++)
        {
            L[i]=a[l+i];
        }
        for(int j=0;j<n2;j++)
        {
            R[j]=a[mid+1+j];
        }

        int i=0,j=0; int
        k=l;

        while(i<n1&& j<n2)
        {
            if(L[i]<R[j])
```

```

        {
            a[k]=L[i];
            i++;
        }
        else
        { a[k]=R[j];
            j++;
        }
        k++;
    }
    while(i<n1)
    { a[k]=L[i];
        k++;
        i++;
    }
    while(j<n2)
    { a[k]=R[j];
        k++;
        j++;
    }
}

void display(int a[])
{
    for(int i=0;i<a.length;i++)
    {
        System.out.println(a[i]);
    }
}

public static void main(String[] args)
{

```

```

MergeSort m=new MergeSort();

Scanner s =new Scanner(System.in);
System.out.println("Enter the limit:"); int
n=s.nextInt(); int a[]=new int[n];

System.out.println("Enter the array:"); for(int i=0;i<n;i++)
{ a[i]=s.nextInt();
}

System.out.println("Before Sorting Array:");


m.display(a);


m.Sort(a,0,a.length-1);

System.out.println("Sorted Array:");

m.display(a);
}
}

```

OUTPUT:

```

Enter the limit:
6
Enter the array:
95
85
65
12
33
54
Sorted Array:
12
33
54
65
85
95

```


Program:

```
import java.util.*;

class LinkedListNew

{

    class Node

    {

        int data;

        Node next;

        Node(int data)

        {

            this.data=data;

        }

    }

    public Node head=null;

    public Node tail=null;

    public void AddNode(int data)

    {

        Node newNode = new Node(data);

        if (head == null)

        {

            head = newNode;
```

```

    }

    else

    {

        tail.next=newNode;

    }

    tail=newNode;

}

public void DeleteFromBeg(int data)

{

    Node temp=head;

    if (temp==null)

    {

        return;

    }

    if (temp!=null)

    {

        head=temp.next;

        tail.next=null;

    }

}

public void DeleteFromEnd(int data)

{

```

```

Node temp=head;

Node prev=null;

if (temp==null)
{
    return;
}

while (temp != null && temp.data!= data)
{
    prev=temp;
    temp=temp.next;
}

if (temp==tail)
{
    tail=prev;
    tail.next=null;
    return;
}

prev.next=temp.next;
}

public void DeleteNode(int data)
{

```

```

Node temp=head;

Node prev=null;

if (temp==null)

{

    return;

}

while (temp != null && temp.data!= data)

{

    prev=temp;

    temp=temp.next;

}

prev.next=temp.next;

}

public void insertAtbeg(int data)

{

    Node newNode= new Node(data);

    Node temp=head;

    if (temp==null)

    {

        return;

    }

    if (temp!=null)

```

```

        {

            newNode.next=head;

            head=newNode;

            tail.next=null;

        }

    }

    public void insertAtEnd(int data)

    {

        Node newNode= new Node(data);

        Node temp=head;

        if (temp==null)

        {

            return;

        }

        if (temp!=null)

        {

            tail.next=newNode;

            tail=newNode;

            tail.next=null;

        }

    }

    public void insertAtSpecific(int nextTo,int data)

```

```

{

    Node newNode= new Node(data);

    Node temp=head;

    if (temp==null)

    {

        return;

    }

    while (temp!=null && temp.data!=nextTo)

    {

        temp=temp.next;

    }

    if (temp==tail)

    {

        tail.next=newNode;

        tail=newNode;

        return;

    }

    newNode.next=temp.next;

    temp.next=newNode;

}

public void display()

{

```

```
        if (head == null)

        {

                System.out.println("empty");

        }

        Node temp=head;

        while (temp != null)

        {

                System.out.print(temp.data+" >> ");

                temp=temp.next;

        }

        System.out.println();

    }

    public static void main(String[] args)

    {

        LinkedListNew list = new LinkedListNew();

        list.AddNode(90);

        list.AddNode(80);

        list.AddNode(70);

        list.AddNode(60);

        list.AddNode(50);

        list.AddNode(40);

    }

}
```

```
System.out.println(" ");

System.out.println("Elements in Linked List are: ");

list.display();

System.out.println("insert at beg");

list.insertAtbeg(30);

list.display();

System.out.println("insert at End");

list.insertAtEnd(100);

list.display();

System.out.println("Insert nextto 70");

list.insertAtSpecific(70,75);

list.display();

System.out.println("delete from begining");

list.DeleteFromBeg(40);

list.display();

System.out.println("delete from End");

list.DeleteFromEnd(100);

list.display();

System.out.println("delete from 75");

list.DeleteNode(75);

list.display();

}
```


}

OUTPUT:

```
Elements in Linked List are:
90 >> 80 >> 70 >> 60 >> 50 >> 40 >>
Insert at beginning
30 >> 90 >> 80 >> 70 >> 60 >> 50 >> 40 >>
Insert at End
30 >> 90 >> 80 >> 70 >> 60 >> 50 >> 40 >> 100 >>
Insert nextto 70
30 >> 90 >> 80 >> 70 >> 75 >> 60 >> 50 >> 40 >> 100 >>
Delete from beginning
90 >> 80 >> 70 >> 75 >> 60 >> 50 >> 40 >> 100 >>
Delete from End
90 >> 80 >> 70 >> 75 >> 60 >> 50 >> 40 >>
Delete 75
90 >> 80 >> 70 >> 60 >> 50 >> 40 >>
```

Program:

```
class DoublyLinked_L{

    Node head;

    class Node{

        int data;

        Node next;

        Node prev;

        Node(int data){

            this .data = data;

        }

    }

    void insert_beg(int data){

        Node new_node = new Node(data);

        new_node.next = head;

        new_node.prev = null;

        if(head!=null){

            head.prev = new_node;

        }

        head = new_node;

        System.out.println("inserted "+ data);

    }

    void insert_end(int data){

        Node new_node = new Node(data);
```

```

new_node.next = null;
if(head == null){
    new_node.prev = null;
    head = new_node;
}
else{
    Node temp = head;
    while(temp.next!=null){
        temp = temp.next;
    }
    temp.next = new_node;
    new_node.prev = temp;
}
System.out.println("inserted "+ data);
}

void insert_after(int nod,int data){
    Node new_node = new Node(data);
    if(head == null){
        new_node.prev = null;
        new_node.next = null;
        head = new_node;
    }
    else{
        Node temp1 = head;
        Node temp2;
        while(temp1.data != nod ){
            if(temp1.next == null){
                System.out.println("Reached last");
                return;
            }
        }
    }
}

```

```

        }
        temp1 = temp1.next;
    }
    temp2 = temp1.next;
    temp1.next = new_node;
    new_node.prev = temp1;
    new_node.next = temp2;
    temp2.prev = new_node;
}
System.out.println("inserted "+ data);
}

void delete_beg(){

    if(head == null){
        System.out.println("List Empty");
        return;
    }
    else{
        Node temp = head;
        if(head.next==null && head.prev==null){
            head = null;
        }
        else{
            head = temp.next;
            head.prev = null;
        }
    }
    System.out.println("Deleted from begining");
}

```

```

void delete_end(){
    if(head == null){
        System.out.println("List Empty");
        return;
    }
    else{
        Node temp = head;
        if(head.next==null && head.prev==null){
            head = null;
        }
        else{
            while(temp.next!=null){
                temp = temp.next;
            }

            temp.prev.next = null;
        }

    }
    System.out.println("Deleted from end");
}

```

```

void delete_node(int data){
    if(head == null){
        System.out.println("List Empty");
    }
    else{
        Node temp = head;
        while(temp.data!= data){

```

```

        if(temp.next == null){
            System.out.println("Reached last node");
            return;
        }
        temp = temp.next;
    }
    if(temp.next == null && temp.prev == null){
        head = null;
        return;
    }
    else{
        if(temp == head){
            head = head.next;
            head.prev = null;
        }
        else if(temp.next == null){
            temp.prev.next = null;
        }
        else{
            temp.prev.next = temp.next;
            temp.next.prev = temp.prev;
        }
    }
}
}

```

```

void display(){
    if(head == null){
        System.out.println("List Empty");
    }
}

```

```

else{
    Node temp = head;
    while(temp!=null){
        System.out.print(temp.data+ " >> ");
        temp = temp.next;

    }
    System.out.println();
}
}

```

```

public static void main(String args[]){

```

```

    DoublyLinked_L dll = new DoublyLinked_L();

```

```

    dll.insert_beg(10);

```

```

    dll.insert_beg(30);

```

```

    dll.insert_beg(26);

```

```

    dll.delete_beg();

```

```

    dll.insert_end(12);

```

```

    dll.display();

```

```

    dll.delete_end();

```

```

    dll.insert_end(88);

```

```

    dll.display();

```

```

    dll.insert_after(3,300);

```

```

    dll.display();

```

```

    dll.insert_after(120,1200);

```

```

    dll.display();

```

```
dll.delete_node(88);  
dll.display();  
  
dll.delete_node(100);  
dll.display();  
}  
}
```

OUTPUT:

```
inserted 10  
inserted 30  
inserted 26  
Deleted from begining  
inserted 12  
30 >> 10 >> 12 >>  
Deleted from end  
inserted 88  
30 >> 10 >> 88 >>  
Reached last  
30 >> 10 >> 88 >>  
Reached last  
30 >> 10 >> 88 >>  
30 >> 10 >>  
Reached last node  
30 >> 10 >>
```


Program:

```
import java.util.*;
class CircularLinkedList
{
    class Node
    {
        int data;
        Node next;

        Node(int data)
        {
            this.data=data;
        }
    }
    public Node head=null;
    public Node tail=null;

    public void AddNode(int data)
    {
        Node newNode = new Node(data);

        if (head == null)
        {
            head = newNode;
        }
        else
        {
            tail.next=newNode;
        }
        tail=newNode;
        tail.next=head;
    }

    public void DeleteFromBeg(int data)
    {
        Node temp=head;
        if (temp==null)
        {
            return;
        }
        if (temp!=null)
        {
            head=temp.next;
            tail.next=head;
        }
    }
}
```

```

    }
    public void DeleteFromEnd(int data)
    {
        Node temp=head;
        Node prev=null;

        if (temp==null)
        {
            return;
        }

        while (temp != null && temp.data!= data)
        {
            prev=temp;
            temp=temp.next;
        }

        if (temp==tail)
        {
            tail=prev;
            tail.next=head;
            return;
        }
        prev.next=temp.next;
    }

```

```

    public void DeleteNode(int data)
    {
        Node temp=head;
        Node prev=null;

        if (temp==null)
        {
            return;
        }

        while (temp != null && temp.data!= data)
        {
            prev=temp;
            temp=temp.next;
        }
        prev.next=temp.next;
    }

```

```

    }
    public void insertAtbeg(int data)
    {
        Node newNode= new Node(data);
        Node temp=head;
        if (temp==null)
        {
            return;
        }
        if (temp!=null)
        {
            newNode.next=head;
            head=newNode;
            tail.next=head;
        }
    }
    public void insertAtEnd(int data)
    {
        Node newNode= new Node(data);
        Node temp=head;
        if (temp==null)
        {
            return;
        }
        if (temp!=null)
        {
            tail.next=newNode;
            tail=newNode;

            tail.next=head;
        }
    }
    public void insertAtSpecific(int nextTo,int data)
    {
        Node newNode= new Node(data);
        Node temp=head;

        if (temp==null)
        {
            return;
        }
        while (temp!=null && temp.data!=nextTo)
        {
            temp=temp.next;
        }
    }

```

```

        if (temp==tail)
        {
            tail.next=newNode;
            tail=newNode;
            tail.next=head;
            return;
        }

        newNode.next=temp.next;
        temp.next=newNode;

    }
    public void display()
    {
        if (head == null)
        {
            System.out.println("empty");
        }
        Node temp=head;
        do
        {
            System.out.print(" "+temp.data);
            temp=temp.next;
        }
        while(temp!=head);
        System.out.println();
    }
    public static void main(String[] args)
    {
        CircularLinkedList list = new CircularLinkedList();
        list.AddNode(10);
        list.AddNode(20);
        list.AddNode(30);
        list.AddNode(40);
        list.AddNode(50);
        list.AddNode(60);

        System.out.println(" ");
        System.out.println("Elements in Linked List are: ");

        list.display();
    }
}

```

```

System.out.println("Insert at beg");
list.insertAtbeg(90);
list.display();
System.out.println("Insert at End");
list.insertAtEnd(80);
list.display();
System.out.println("Insert nextto 40");
list.insertAtSpecific(40,100);
list.display();

System.out.println("Delete from begining");
list.DeleteFromBeg(90);
list.display();

System.out.println("Delete from End");
list.DeleteFromEnd(80);
list.display();

System.out.println("Delete from Specific Node");
list.DeleteNode(50);
list.display();

```

```

}

```

```

}

```

OUTPUT:

```

C:\Users\ASUS\Downloads>java CircularLinkedList

Elements in Linked List are:
 10 20 30 40 50 60
Insert at beg
 90 10 20 30 40 50 60
Insert at End
 90 10 20 30 40 50 60 80
Insert nextto 40
 90 10 20 30 40 100 50 60 80
Delete from begining
 10 20 30 40 100 50 60 80
Delete from End
 10 20 30 40 100 50 60
Delete from Specific Node
 10 20 30 40 100 60

```

Program:

```
class RevLinkedListNew
{
    class Node
    {
        int data;
        Node next;

        Node(int data)
        {
            this.data=data;
        }
    }
    public Node head=null;
    public Node tail=null;

    public void AddNode(int data)
    {
        Node newNode = new Node(data);

        if (head == null)
        {
            head = newNode;
        }
        else
        {
            tail.next=newNode;
        }
        tail=newNode;
    }
    public void display()
    {
        if (head == null)
        {
            System.out.println("empty");
        }
        Node temp=head;
        while (temp != null)
        {
            System.out.println(temp.data);
            temp=temp.next;
        }
    }
    public void reverse()
```

```

    {
Node pointer = head;
    Node previous = null, current = null;

    while (pointer != null)
    {
        current = pointer;
        pointer = pointer.next;
        current.next = previous;
        previous = current;
        head = current;
    }
}

public static void main(String[] args)
{
    RevLinkedListNew list = new RevLinkedListNew();
    list.AddNode(90);
    list.AddNode(80);
    list.AddNode(70);
    list.AddNode(60);
    list.AddNode(50);
    list.AddNode(40);
    System.out.println("Elements in Linked List are:");
    list.display();
    System.out.println();
    System.out.println("Reversed Linked List are:");
    list.reverse();
    list.display();
}
}

```

OUTPUT:

```

C:\Users\ASUS\Downloads>java RevLinkedListNew
Elements in Linked List are:
90
80
70
60
50
40

Reversed Linked List are:
40
50
60
70
80
90

```

Program:

```
import java.util.*;

class StackLink
{
    StackNode top;

    class StackNode
    {
        int data; StackNode
        next;

        StackNode(int data)
        {
            this.data=data;
        }
    }

    void push(int data)
    {
        StackNode newNode=new StackNode(data); if
        (top==null)
        {
            top=newNode;
        }
        else
        {
            StackNode temp=top; top=newNode;
            newNode.next=temp;
        }
    }

    void pop()
    {
        if (top==null)
        {
```



```

        System.out.println("Stack Empty");
    }
    else
    {
        int popped=top.data; top=top.next;
        System.out.println("popped"+popped);
    }
}
void peek()
{
    if (top==null)
    {
        System.out.println("Stack Empty");
    }
    else
    {
        System.out.println("peeked"+top.data);
    }
}
public void display()
{
    if (top == null)
    {
        System.out.println("empty");
    }
    StackNode temp=top;
    while (temp != null)
    {
        System.out.println(temp.data);
        temp=temp.next;
    }
}

```

```

    }
}

public static void main(String [] args)
{
    StackLink sc=new StackLink();
    Scanner s= new Scanner(System.in);
    System.out.println("Enter the number of elemet to
insert");
    int n = s.nextInt();
    System.out.println("Enter the elements");
    for(int i=0;i<n;i++)
    { sc.push(s.nextInt());
    }
    System.out.println("Stack Elements");
    sc.display();
    System.out.println();
    sc.peek();
    sc.pop();
    sc.peek();
    sc.pop();
    sc.peek();
    sc.pop();
    System.out.println("Current Stack Elements");
    sc.display();
}
}

```

OUTPUT:

```
Enter the number of elements to insert
6
Enter the elements
45
55
24
96
62
75
Stack Elements:
75 62 96 24 55 45

Current top : 75
Deleted element : 75
Current top : 62
Deleted element : 62
Current top : 96
Deleted element : 96
Current Stack Elements:
24 55 45
```

Program:

```
import java.util.*;
class QueueLink
{
    class QueueNode
    {
        int data;
        QueueNode next;
        QueueNode(int data)
        {
            this.data=data;
        }
    }
    QueueNode front;
    QueueNode rear;
    public void Enqueue(int data)
    {
        QueueNode NewNode=new QueueNode(data);
        if (front==null)
        {
            front=NewNode;
        }
        else
        {
            rear.next=NewNode;
        }
        rear=NewNode;
    }
    public void Dequeue()
    {
        if (front==null)
        {
            System.out.println("Queue empty");
        }
        else
        {
            System.out.println("popped"+front.data);
            front=front.next;
        }
    }
    public void display()
    {
        if (front == null)
        {
            System.out.println("empty");
        }
    }
}
```

```

    }
    QueueNode temp=front;
    while (temp != null)
    {
        System.out.println(temp.data);
        temp=temp.next;
    }
}
public static void main(String[] args)
{
    QueueLink que=new QueueLink();

    Scanner s= new Scanner(System.in);
    System.out.println("Enter the number of elemet to insert");
    int n = s.nextInt();
    System.out.println("Enter the elements");
    for(int i=0;i<n;i++)
    {
        que.Enqueue(s.nextInt());
    }
    que.Dequeue();
    que.Dequeue();
    System.out.println("Current Queue Elements");
    que.display();
}
}

```

OUTPUT:

```

C:\Users\ASUS\Downloads>java QueueLink
Enter the number of elemet to insert
5
Enter the elements
6
8
4
7
2
popped6
popped8
Current Queue Elements
4
7
2

```

Program:

```
class Node{
    int data;
    Node left;
    Node right;

    Node(int data){
        this.data = data;
        left = null;
        right = null;
    }
}

class BinarySearchT{

    Node root;

    BinarySearchT(){
        this.root = null;
    }

    public Node search(Node root, int d){

        if(root == null || root.data == d){
            return root;
        }
        if(root.data < d){
            return search(root.right, d);
        }
        return search(root.left, d);
    }
}
```

```
}
```

```
public void insert(int d){
```

```
    Node new_node = new Node(d);
```

```
    Node temp = this.root;
```

```
    Node y = null;
```

```
    while(temp!=null){
```

```
        y = temp;
```

```
        if(d<temp.data){
```

```
            temp = temp.left;
```

```
        }
```

```
        else{
```

```
            temp = temp.right;
```

```
        }
```

```
    }
```

```
    if(y == null){
```

```
        this.root = new_node;
```

```
    }
```

```
    else if(d<y.data){
```

```
        y.left = new_node;
```

```
    }
```

```
    else{
```

```
        y.right = new_node;
```

```
    }
```

```
}
```

```
public boolean delete (int d){
```

```
Node curr = this.root;
Node parent = this.root;
boolean isLeftChild = false;
```

```
while(curr.data!=d){
    parent = curr;
    if(curr.data>d){
        isLeftChild = true;
        curr = curr.left;
    }
    else{
        isLeftChild = false;
        curr = curr.right;
    }
    if(curr == null){
        return false;
    }
}
```

```
if(curr.left == null && curr.right == null){
    if(curr == this.root){
        this.root = null;
    }
    if(isLeftChild){
        parent.left = null;
    }
    else{
        parent.right = null;
    }
}
```



```
}
```

```
else if(curr.right == null){  
    if(curr == this.root){  
        this.root = curr.left;  
    }  
    else if(isLeftChild){  
        parent.left = curr.left;  
    }  
    else{  
        parent.right = curr.left;  
    }  
}
```

```
else if(curr.left == null){  
    if(curr == this.root){  
        this.root = curr.right;  
    }  
    else if(isLeftChild){  
        parent.left = curr.right;  
    }  
    else{  
        parent.right = curr.right;  
    }  
}
```

```
else if(curr.left != null && curr.right!=null){  
    Node successor = getSuccessor(curr);  
    if(curr == root){  
        this.root = successor;
```

```

    }
    else if(isLeftChild){
        parent.left = successor;
    }
    else{
        parent.right = successor;
    }
    successor.left = curr.left;
}

return true;
}

public Node getSuccessor(Node del_nod){

    Node successor = null;
    Node successorParent = null;
    Node current = del_nod.right;

    while(current!=null){
        successorParent = successor;
        successor = current;
        current = current.left;
    }

    if(successor != del_nod.right){
        successorParent.left = successor.right;
        successor.right = del_nod.right;
    }

    return successor;
}

```

```
}
```

```
public void inOrder(Node root){  
    if(root != null){  
        inOrder(root.left);  
        System.out.print(root.data + " ");  
        inOrder(root.right);  
    }  
}
```

```
public static void main(String args[]){  
  
    BinarySearchT bst = new BinarySearchT();  
  
    bst.insert(45);  
    bst.insert(10);  
    bst.insert(7);  
    bst.insert(12);  
    bst.insert(90);  
    bst.insert(50); //7 10 12 45 50 90  
    System.out.println("The inorder of created BST: ");  
    bst.inOrder(bst.root);  
    System.out.println();  
  
    System.out.println("\nSearch node 90 in BST: ");  
    if(bst.search(bst.root,90) != null){  
        System.out.println("Node 90 is present ");  
    }  
    else{
```

```
        System.out.println("Node 90 is not present ");
    }

    System.out.println();

    bst.delete(12);
    System.out.println("After deletion of 12:");
    bst.inOrder(bst.root);

    System.out.println();

    System.out.println("\nSearch node 12 in BST: ");
    if(bst.search(bst.root,12) != null){
        System.out.println("Node 12 is present ");
    }
    else{
        System.out.println("Node 12 is not present ");
    }
    System.out.println();

    bst.delete(90);
    System.out.println("After deletion of 90:");
    bst.inOrder(bst.root);
    System.out.println();
    bst.delete(45);
    System.out.println("After deletion of 45:");
    bst.inOrder(bst.root);
}
}
```

OUTPUT:

```
The inorder of created BST:  
7 10 12 45 50 90
```

```
Search node 90 in BST:  
Node 90 is present
```

```
After deletion of 12:  
7 10 45 50 90
```

```
Search node 12 in BST:  
Node 12 is not present
```

```
After deletion of 90:  
7 10 45 50
```

```
After deletion of 45:  
7 10 50
```

Program:

```
import java.util.*;

public class InfixToPostFix {

    static int precedence(char c){

        switch (c){

            case '+':

            case '-':

                return 1;

            case '*':

            case '/':

                return 2;

            case '^':

                return 3;

        }

        return -1;

    }

    static String infixToPostFix(String expression){

        String result = "";

        Stack<Character> stack = new Stack<>();

        for (int i = 0; i < expression.length() ; i++) {

            char c = expression.charAt(i);

            if(precedence(c)>0){

                while(stack.isEmpty()==false && precedence(stack.peek())>=precedence(c)){

                    result += stack.pop();

                }

                stack.push(c);

            }else if(c=='('){

                char x = stack.pop();

                while(x!='('){

                    result += x;

                }

            }

        }

        return result + stack.pop();

    }

}
```

```

        x = stack.pop();
    }
    }else if(c=='('){
        stack.push(c);
    }else{
        result += c;
    }
}
for (int i = 0; i <=stack.size() ; i++) {
    result += stack.pop();
}
return result;
}

public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    System.out.println("Enter the expression: ");
    String exp = myObj.nextLine();
    System.out.println("Infix Expression: " + exp);
    System.out.println("Postfix Expression: " + infixToPostFix(exp));
}
}

```

Output

```

Enter the expression:
(A+B*(C+D))/(F+D*E)
Infix Expression: (A+B*(C+D))/(F+D*E)
Postfix Expression: ABCD+*+FDE*+/>

```

Program:

```
class Node{
    int data;
    Node left;
    Node right;

    Node(int data){
        this.data = data;
        left = null;
        right = null;
    }
}

class BinarySearchTTraversal{

    Node root;

    BinarySearchTTraversal(){
        this.root = null;
    }

    public void insert(int d){
        Node new_node = new Node(d);

        Node temp = this.root;
        Node y = null;

        while(temp!=null){
            y = temp;
            if(d<temp.data){
```



```

        temp = temp.left;
    }
    else{
        temp = temp.right;
    }
}

if(y == null){
    this.root = new_node;
}
else if(d<y.data){
    y.left = new_node;
}
else{
    y.right = new_node;
}
}

public void inOrder(Node root){
    if(root != null){
        inOrder(root.left);
        System.out.print(root.data + " ");
        inOrder(root.right);
    }
}

public void preOrder(Node root){
    if(root!=null){
        System.out.print(root.data + " ");

```

```

        preOrder(root.left);
        preOrder(root.right);

    }
}

public void postOrder(Node root){
    if(root!= null){
        postOrder(root.left);
        postOrder(root.right);
        System.out.print(root.data + " ");
    }
}

public static void main(String args[]){

    BinarySearchTTraversal bst = new BinarySearchTTraversal();

    bst.insert(52);
    bst.insert(15);
    bst.insert(56);
    bst.insert(9);
    bst.insert(16);
    bst.insert(54);
    bst.insert(3);
    bst.insert(10);
    bst.insert(61);

    System.out.println("Inorder Traversal of created BST: ");
    bst.inOrder(bst.root);
    System.out.println();
}

```

```
        System.out.println("Preorder Traversal of created BST: ");
        bst.preOrder(bst.root);
        System.out.println();

        System.out.println("Postorder Traversal of created BST: ");
        bst.postOrder(bst.root);
    }
}
```

OUTPUT:

```
Inorder Traversal of created BST:
3 9 10 15 16 52 54 56 61
Preorder Traversal of created BST:
52 15 9 3 10 16 56 54 61
Postorder Traversal of created BST:
3 10 9 16 15 54 61 56 52
```