LAB PROGRAM -1

1) Write a java program to implement PUSH and POP operations on Stack using array method.

AIM: Write a java program to implement PUSH and POP operations on Stack using array method.

PROGRAM

```
import java.io.*;
class StackA
{
    int maxsize=50;
    int st[];
    int top;
    StackA()
    {
        st=new int[maxsize];
        top=-1;
    }
    public void push(int j)
    {
        if(top==maxsize-1)
        {
            System.out.println("Stack Overflow");
            return;
        }
        st[++top]=j;
        System.out.println(j+" Item pushed into the stack");
    }
    public void pop()
    {
        if(top==-1)
        {
            System.out.println("\nStack is empty");
            return;
        }
```

```
        System.out.println("\n"+st[top]+"Item Deleted from Stack");
        top=top-1;
    }
    public void show()
    {
        int temp;
        if(top==-1)
        {
            System.out.println("\nStack is empty");
            return;
        }
        temp=top;
        System.out.println("Stack item are");
        for(int i=temp;i>=0;i--)
            System.out.println(st[i]);
    }
}
class StackArray
{
    public static void main(String args[])throws Exception
    {
        int ch;
        BufferedReader br=new BufferedReader(new
            InputStreamReader(System.in));
        StackA sa=new StackA();
        do
        {
            System.out.println("1.PUSH \t 2.POP \t 3.SHOW \t
                4.EXIT \n Enter your choice");
            ch=Integer.parseInt(br.readLine());
            switch(ch)
            {
                case 1:
                 System.out.println("Enter an item :");
                int item=Integer.parseInt(br.readLine());
                sa.push(item);
                break;
              case 2:
                sa.pop();
                break;
              case 3:
                sa.show();
                break;
              case 4:
                System.out.println("Execution is Over ");
                System.exit(0);
                break;
              default:
                System.out.println("Wrong Choice");
          }
        }while(ch!=4);
    }
}
```

TEST DATA AND OUTPUT

```
C:\java>java StackArray
1.PUSH     2.POP     3.SHOW     4.EXIT
Enter your choice : 1
Enter an item : 10
10 Item pushed into the stack
1.PUSH     2.POP     3.SHOW     4.EXIT
Enter your choice : 1
Enter an item : 20
20 Item pushed into the stack
1.PUSH     2.POP     3.SHOW     4.EXIT
Enter your choice : 1
Enter an item : 30
30 Item pushed into the stack
1.PUSH     2.POP     3.SHOW     4.EXIT
```

```
Enter your choice : 3
Stack item are
30
20
10
1.PUSH     2.POP     3.SHOW     4.EXIT
Enter your choice : 2
30 Item Deleted from Stack
1.PUSH     2.POP     3.SHOW     4.EXIT
Enter your choice : 3
Stack item are
20
10
1.PUSH     2.POP     3.SHOW     4.EXIT
Enter your choice : 4
Execution is Over
```

LAB PROGRAM -2

2) Write a java Program to implement insert and delete operations on Queue using array method.

AIM : Write a java Program to implement insert and delete operations on Queue using array method.

PROGRAM

```
import java.io.*;
class Queue
{
     int maxsize=50;
     int[] queArray;
     int front;
     int rear;
     int data;
     Queue()
     {
          queArray=new int[maxsize];
          front=-1;
          rear=-1;
          data=0;
    }
    public void insert(int j)
    {
         if(rear==maxsize-1)
         {
              System.out.println("Queue is overflow");
         }
         else
         {
              rear++;
              queArray[rear]=j;
              System.out.println(j+"item inserted into queue");
              if(front==-1)
              front=0;
```

```
          }
    }
    public void delete()
    {
         if(front==-1||rear==-1)
              System.out.println("Queue is underflow");
         else
         {
              System.out.println("Deleted item is:"+
                   queArray[front]);
              front++;
              if(rear+1==front)
              rear=front=-1;
         }
    }
    public void show()
    {
         int temp;
         if(front==-1||rear==-1)
         {
              System.out.println("\nQueue is empty");
              return;
         }
         System.out.print("Front->");
         for(int i=front;i<=rear;i++)
              System.out.print(queArray[i]+"->");
         System.out.println("Rear");
    }
}
class QueueArray
{
    public static void main(String args[])throws Exception
    {
         int ch;
```

```
        BufferedReader br=new BufferedReader(new
            InputStreamReader(System.in));
        Queue q=new Queue();
        do
        {
            System.out.println("1.Insert \t 2.Delete \t
                3.Show \t 4.Exit\n Enter your choice");
            ch=Integer.parseInt(br.readLine());
            switch(ch)
            {
                case 1:
                  System.out.println("\n Enter item:");
                  int item=Integer.parseInt(br.readLine());
                  q.insert(item);
                  break;
                case 2:
                  q.delete();
                  break;
                case 3:
                  q.show();
                  break;
                case 4:
                  System.out.println("Execution is Over");
                  System.exit(0);
                  break;
                default:
                  System.out.println("Wrong Choice");
            }
        }while(ch!=4);
    }
}
```

TEST DATA AND OUTPUT

```
C:\java>java QueueArray
1.Insert        2.Delete        3.Show        4.Exit
```

```
Enter your choice : 3
Queue is empty
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 1
Enter item : 10
10 item inserted into queue
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 1
Enter item : 20
20 item inserted into queue
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 1
Enter item : 30
30 item inserted into queue
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 3
Front->10->20->30->Rear
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 2
Deleted item is :10
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 3
Front->20->30->Rear
1.Insert        2.Delete        3.Show        4.Exit
Enter your choice : 4
Execution Over
```

LAB PROGRAM - 3

3) Write a java program to create, insert, delete and display operations on single linked list.

AIM : Write a java program to create, insert, delete and display operations on single linked list.

PROGRAM

```
import java.io.*;
class Link
{
        public int idata;
        public Link next;
        public Link(int id)
        {
                idata=id;
                next=null;
        }
        public void displayLink()
        {
                System.out.print("->"+idata);
        }
}
class LinkedList
{
        private Link first,last;
        static int count=0;
        public LinkedList()
        {
                first=null;
                last=null;
        }
        public boolean isEmpty()
        {
                return(first==null);
        }
        public void insertFirst(int id)
```

```
        {
                Link newLink=new Link(id);
                if(isEmpty())
                        first=newLink;
                else
                        last.next=newLink;
                        last=newLink;
        }
        public Link deleteFirst()
        {
                if(first!=null)
                {
                        Link temp=first;
                        first=first.next;
                        return temp;
                }
                else
                        return(null);
        }
        public void displayList()
        {
                if(first!=null)
                {
                        System.out.print("List(first->last):");
                        Link current=first;
                        while(current!=null)
                        {
                                current.displayLink();
                                current=current.next;
                        }
                        System.out.println(" ");
                }
                else
                        System.out.println("List is empty");
```

```
      }
}
class SingleLinkedList
{
     public static void main(String args[])throws Exception
     {
          LinkedList theList=new LinkedList();
          boolean flag=false;
          int ch;
          int value=0;
          BufferedReader br=new BufferedReader(new
               InputStreamReader(System.in));
      do
      {
          System.out.println("\n 1.Create\t 2.Insert \t
               3.Display \t 4.Delete \t 5.Exit \n Enter your
               choice:\n");
          ch=Integer.parseInt(br.readLine());
          switch(ch)
          {
          case 1:
            flag=theList.isEmpty();
            if(flag==true)
            {
              System.out.println("LinkedList is created select
                insert to insert data");
              flag=true;
             }
             else
              System.out.println("LinkedList is Already created!!");
            break;
            case 2:
              if(flag==false)
              System.out.println("created Linked List First!!");
```

```
else
{
     System.out.println("enter item to insert:");
     int n=Integer.parseInt(br.readLine());
     theList.insertFirst(n);
     value=1;
}
break;
case 3:
  if(flag==false)
     System.out.println("Create Linked List first!!!");
  else
     theList.displayList();
break;
case 4:
  if(flag==false)
     System.out.println("Create LinkedListFirst!!!");
     if(value==0)
       System.out.println("List is empty");
     else
     {
       Link  aLink=theList.deleteFirst();
       if(aLink!=null)
       {
          System.out.println("Deleted item");
          aLink.displayLink();
       }
       else
       {
          System.out.println("All items deleted
               create List again");
          value=0;
       }
     }
```

```
            break;
            case 5:
                System.out.println("Execution is Over ");
                System.exit(0);
                break;
            default :
                System.out.println("Wrong choice!!!");
                break;
            }
        }while(ch!=5);
    }
}
```

TEST DATA AND OUTPUT

```
C:\java>java SingleLinkedList
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 1
Linked List is created select insert to insert data
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 2
enter item to insert: 10
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 2
enter item to insert: 20
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 2
enter item to insert: 30
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 3
List(first->last):->10->20->30
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 4
Deleted item ->10
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 3
List(first->last):->20->30
1.Create      2.Insert      3.Display      4.Delete      5.Exit
Enter your choice: 5
Execution is Over
```

LAB PROGRAM - 4

4) Write a java program to implement Bubble Sort.

AIM : Write a java program to implement Bubble Sort.

PROGRAM

```
import java.io.*;
class ArrayBubble
{
    private int[] a;
    private int n;
    public ArrayBubble(int max)
    {
        a=new int[max];
        n=0;
    }
    public void insert(int value)
    {
        a[n]=value;
        n++;
    }
    public void display()
    {
        for(int j=0;j<n;j++)
            System.out.print(a[j]+" \t ");
    }
    public void bubblesort()
    {
        int out,in;
        for(out=n-1;out>0;out--)
            for(in=0;in<out;in++)
                swap(in,in+1);
    }
    private void swap(int one,int two)
    {
        if(a[one]>a[two])
        {
              int temp=a[one];
              a[one]=a[two];
              a[two]=temp;
        }
    }
}
class BubbleSort
{
    public static void main(String args[]) throws Exception
    {
        BufferedReader br=new BufferedReader(new
          InputStreamReader(System.in));
        System.out.println("Enter how many elements to sort:");
        int size=Integer.parseInt(br.readLine());
        ArrayBubble ob=new ArrayBubble(size);
        for(int i=1;i<=size;i++)
        {
           System.out.println("Enter element : ");
           ob.insert(Integer.parseInt(br.readLine()));
        }
        System.out.println("Elements before sorting");
        ob.display();
        ob.bubblesort();
        System.out.println("\n Elements after sorting");
        ob.display();
    }
}
```

TEST DATA AND OUTPUT

```
C:\java>java BubbleSort
Enter how many elements to sort :
6
Enter element : 66
Enter element : 55
```

```
Enter element : 22
Enter element : 99
Enter element : 77
Enter element : 11
Elements before sorting
66    55    22    99    77    11
Elements after sorting
11    22    55    66    77    99
```

LAB PROGRAM - 5

5) Write a java program to implement Insertion Sort.

AIM : Write a java program to implement Insertion Sort.

PROGRAM

```
import java.util.Scanner;
public class InsertionSort
{
    public static void sort( int arr[] )
    {
        int N = arr.length;
        int i, j, temp;
        for (i=1; i<N; i++)
        {
            j = i;
            temp = arr[i];
            while (j>0 && temp<arr[j-1])
            {
                arr[j] = arr[j-1];
                j = j-1;
            }
            arr[j] = temp;
        }
  }
  public static void main(String[] args)
  {
      Scanner scan = new Scanner( System.in );
      System.out.println("Insertion Sort Test\n");
      int n, i;
      System.out.println("Enter number of integer
          elements");
      n = scan.nextInt();
      int arr[] = new int[ n ];
      System.out.println("\n Enter "+ n +" integer
          elements");
```

```
      for (i = 0; i < n; i++)
          arr[i] = scan.nextInt();
      System.out.println("\n Elements before sorting ");
      for (i = 0; i < n; i++)
          System.out.print(arr[i]+" ");
      sort(arr);
      System.out.println("\nElements after sorting ");
      for (i = 0; i < n; i++)
          System.out.print(arr[i]+" ");
      System.out.println();
  }
}
```

TEST DATA AND OUTPUT

```
C:\java>java InsertionSort
Insertion Sort Test
Enter number of integer elements : 8
Enter 8 integer elements
56
25
34
66
88
99
11
22
Elements before sorting
56 25 34 66 88 99 11 22
Elements after sorting
11 22 25 34 56 66 88 99
```

LAB PROGRAM - 6

6) Write a java program to implement Selection Sort.

AIM : Write a java program to implement Selection Sort.

PROGRAM

```
import java.util.Scanner;
public class SelectionSort
{
    public static void  selectionSort(int[] arr)
    {
        int i, j, minIndex, tmp;
        int n = arr.length;
        for (i = 0; i < n - 1; i++)
        {
            minIndex = i;
            for (j = i + 1; j < n; j++)
                if (arr[j] < arr[minIndex])
                    minIndex = j;
            if (minIndex != i)
            {
                tmp = arr[i];
                arr[i] = arr[minIndex];
                arr[minIndex] = tmp;
            }
        }
}
public static void main(String a[])
{
  Scanner scan = new Scanner( System.in );
  int i,n;
  System.out.println("Enter No. of elements to sort : ");
n= scan.nextInt();
int  arr[] =new int[n];
System.out.println("Enter "+n+"elements into the array");
   for(i=0;i<arr.length;i++)
```

```
    {
        arr[i]= scan.nextInt();
    }
    System.out.println(" Selection Sort\n\n");
    System.out.println("Elements before sorting\n");
    for(i = 0; i < arr.length; i++)
        System.out.print( arr[i]+"  ");
    selectionSort(arr);
    System.out.println("\n Elements after sorting\n");
    for(i = 0; i <arr.length; i++)
        System.out.print(arr[i]+"  ");
  }
}
```

TEST DATA AND OUTPUT

```
C:\java>java SelectionSort
Enter No. of elements to sort :
8
Enter 8 elements into the array
23
35
45
11
26
10
5
8
Selection Sort
Elements before sorting
23 35 45 11 26 10 5 8
Elements after sorting
5 8 10 11 23 26 35 45
```

LAB PROGRAM - 7

7)Write a java program to implement Linear Search.

AIM : Write a java program to implement Linear Search.

PROGRAM

```
import java.util.Scanner;
class LinearSearch
{
      public static void main(String args[])
      {
            int num,i, se,array[];
            Scanner input = new Scanner(System.in);
            System.out.println("Enter number of elements:");
            num = input.nextInt();
            array = new int[num];
            System.out.println("Enter " + num + " integers");
            for (i= 0; i < num; i++)
            array[i] = input.nextInt();
            System.out.println("ENTER THE NUMBER TO BE SEARCHED :");
            se = input.nextInt();
            for (i = 0; i < num; i++)
            {
                  if (array[i] == se)
                  {
                    System.out.println(se+"is present at location"+(i+1)
                    break;
                  }
            }
            if (i == num)
                  System.out.println(se + " doesn't exist in
            array.");
      }
}
```

TEST DATA AND OUTPUT

1)

```
C:\java>java LinearSearch
```

```
Enter number of elements: 5
Enter 5 Integers
90
20
30
80
70
ENTER THE NUMBER TO BE SEARCHED
70
70 is present at location 5
```

2)

```
C:\java>java LinearSearch
Enter number of elements: 5
Enter 5 Integers
8
40
60
10
50
ENTER THE NUMBER TO BE SEARCHE
90
90 doesn't exist in array
```

LAB PROGRAM - 8

8)Write a java program to implement Binary Search.

AIM : To implement Binary Search.

PROGRAM

```
import java.util.Scanner;
class BinarySearch
{
      public static void main(String args[])
      {
           int i,n;
           Scanner scan = new Scanner( System.in );
           System.out.println("ENTER HOW MANY NUMBERS");
           n = scan.nextInt();
           int a[]=new int[n];
           System.out.println("ENTER "+n+"NUMBERS FOR THE ARRAY");
           for(i=0;i<a.length;i++)
                a[i]=scan.nextInt();
           System.out.println("CONTENTS OF THE ARRAY IS");
           for(i=0;i<a.length;i++)
                System.out.println(a[i]);
           System.out.println("ENTER THE NUMBER TO BE SEARCHED");
           int se=scan.nextInt();
           System.out.println("NUMBER TO BE SEARCHED IS "+se);
           int p=-1,mid,l=0,u=a.length-1;
           while(l<=u)
           {
                mid=(l+u)/2;
                     if(a[mid]==se)
                     {
                          p=mid;
                          break;
                     }
                     else if(se> a[mid])
                     {
                          l=mid+1;
```

```
                     }
                     else if(se<a[mid])
                     {
                          u=mid-1;
                     }
           }
           if(p==-1)
             System.out.println("NUMBER DOES NOT EXIST IN THE ARRAY");
           else
             System.out.println("NUMBER EXISTS AT THE INDEX "+(p+1));
      }
}
```

TEST DATA AND OUTPUT

```
1)   C:\java>java BinarySearch
     ENTER HOW MANY NUMBERS  : 5
     ENTER 5 NUMBERS FOR THE ARRAY
     10
     20
     60
     80
     70
     CONTENTS OF THE ARRAY IS
     10     20     60     80     70
     ENTER THE NUMBER TO BE SEARCHED
     80
     NUMBER TO BE SEARCHED IS 80
     NUMBER EXISTS AT THE INDEX 4
2)   C:\java>java BinarySearch
     ENTER HOW MANY NUMBERS
     3
     ENTER 3 NUMBERS FOR THE ARRAY
     15
     25
     55
     CONTENTS OF ARRAY ARE
     15     25     55
     ENTER THE NUMBER TO BE SEARCHED
     45
     NUMBER TO BE SEARCHED IS 45
     NUMBER DOES NOT EXIST IN THE ARRAY
```

LAB PROGRAM - 9

9) Write a java program to perform graph traversing using DFS.

AIM: To perform graph traversing using DFS.

PROGRAM

```
import java.io.*;
import java.util.*;
class Graph
{
    private int V;
    private LinkedList<Integer> adj[];
    Graph(int v)
    {
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList();
    }
    void addEdge(int v, int w)
    {
        adj[v].add(w);  // Add w to v's list.
    }
    void DFSUtil(int v,boolean visited[])
    {
        visited[v] = true;
        Systε .out.print(v+" ");
        Iterator<Integer> i = adj[v].listIterator();
        while (i.hasNext())
        {
            int n = i.next();
            if (!visited[n])
                DFSUtil(n, visited);
        }
    }
    void DFS(int v)
```

```
    {
        boolean visited[] = new boolean[V];
        DFSUtil(v, visited);
    }
    public static void main(String args[])
    {
        Graph g = new Graph(4);
        g.addEdge(0, 1);
        g.addEdge(1, 2);
        g.addEdge(1, 3);
        g.addEdge(2, 0);
        g.addEdge(2, 3);
        g.addEdge(3, 3);
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the vertex to start DFS :");
        int sv = input.nextInt();
        if(sv<0 || sv>3)
            System.out.println("Not a Valid Vertex");
        else
        {
            System.out.println("Graph Traversing using DFS: \n");
            g.DFS(sv);
        }
    }
}
```

TEST DATA AND OUTPUT

1) C:\>java Graph

```
Enter the vertex to start DFS :
1
Graph Traversing using DFS:
1 2 0 3
```

2) C:\>java Graph

```
Enter the vertex to start DFS :
2
Graph Traversing using DFS:
2 0 1 3
```

3) C:\>java Graph

```
Enter the vertex to start DFS :
6
Not a Valid Vertex
```

LAB PROGRAM - 10

10) Write a java program to perform graph traversing using BFS.

***AIM:** To perform graph traversing using BFS.*

PROGRAM

```
import java.util.*;
public class BFS
{
    private int n;
    private LinkedList<Integer> adjList[];
    private Queue<Integer> q = new LinkedList();
    public void makeGraph(int no)
    {
        n = no;
        adjList =  new LinkedList[no];
        int i;
        for (i= 0; i < no; i++)
        {
            adjList[i] = new LinkedList();
        }
    }
   public void addEdgeToGraph(int u, int v)
   {
       adjList[u].add(v);
   }

   public void BFtraversal(int v, boolean[] visited)
   {
       q.add(v);
       visited[v]  =  true;
       int k;
       while( !q.isEmpty() )
       {
           k = q.remove();
           System.out.print( k +" ");
```

```
            Iterator<Integer> i = adjList[k].listIterator();
            int j;
            while( i.hasNext() )
            {
                j = i.next();
                if( visited[j] != true )
                {
                        q.add(j);
                        visited[j] = true;

                }
            }
        }
    }
    public void BFsearch(int v)
    {
        boolean visited[] = new boolean[n];
        BFtraversal(v, visited);
        for ( int i = 0; i < n; i++ )
        {
            if( visited[i] != true )
            {
                BFtraversal(i, visited);
            }
        }
    }
    public static void main(String args[])
    {
        BFS obj = new BFS();
        obj.makeGraph(4);
        obj.addEdgeToGraph(0, 1);
        obj.addEdgeToGraph(0, 2);
        obj.addEdgeToGraph(1, 3);
        obj.addEdgeToGraph(1, 2);
        obj.addEdgeToGraph(2, 3);
```

```
        obj.addEdgeToGraph(3, 3);
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the vertex to start DFS:");
        int sv = input.nextInt();
        if(sv<0 || sv>3)
          System.out.println("Not a Valid Vertex");
        else
        {
          System.out.println("Graph Traversing using BFS: \n");
          obj.BFsearch(sv);
        }

    }
}
```

TEST DATA AND OUTPUT

1) C:\>java BFS

```
BFS of graph is:
Enter the vertex to start DFS :
0
Graph Traversing using DFS:
0 1 2 3
```

2) C:\>java BFS

```
BFS of graph is:
Enter the vertex to start DFS :
1
Graph Traversing using DFS:
1 3 2 0
```

3) C:\>java BFS

```
BFS of graph is:
Enter the vertex to start DFS :
4
Not a Valid Vertex
```