ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Affiliated to JNTUK, Kakinada and Approved by NBA & NAAC)

Recognized by UGC under section 2(f) and 12(B) of UGC act 1956

Aditya Nagar, ADB Road, Surampalem – 533 437

LABORATORY RECORD

III B.Tech I Sem ECE

STUDENT MANUAL

DATA STRUCTURES USING JAVA LAB

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

2024-2025

VISION & MISSION OF THE INSTITUTE

VISION

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Value based instruction and to emerge as a premiere institute.

MISSION

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative Research and development
- Industry Institute Interaction
- Empowered Manpower

VISION & MISSION OF THE DEPARTMENT

VISION

To be a centre of excellence and recommended for electronics & communication engineering education and research.

MISSION

M1: enlighten the graduates in the basic concepts underlying the principles of analog and digital electronics, communication systems and advanced technologies.

M2:provide state of the art infrastructure and research facilities.

M3: Organizing industrial programs and social activities in collaboration with industries, NSS to disseminate knowledge.

ADITYA COLLEGE OF ENGINEERING& TECHNOLOGY

(Affiliated to JNTUK, Kakinada and Approved by NBA & NAAC)

Recognized by UGC under section 2(f) and 12(B) of UGC act 1956

Aditya Nagar, ADB Road, Surampalem – 533 437

III Year - I Semester		L	T	P	C
		0	0	3	1.5
DATA STRUCTURE USING JAVA LAB					

List of the Experiments:

1. Write Java programs that use both recursive and non-recursive functions for implementing the Linear Search
2. Write Java programs that use both recursive and non-recursive functions for implementing the Binary Search
3. Write a Java program to implement the List ADT using arrays
4. Write a Java program to implement the List ADT using Linked list
5. Write Java program to implement the stack ADT using array
6. Write Java program to implement the Queue ADT using array
7. Write a java program that reads an infix expression, converts the expression to postfix form and then evaluates the postfix expression (use stack ADT).
8. Write a Java program to implement the Stack ADT using a singly linked list
9. Write a Java program to implement the Queue ADT using a singly linked list.
10. Write Java programs that use recursive and non-recursive functions to traverse the given binary tree in

    a) Preorder
    b) Inorder
    c) Postorder.
11. Write Java programs for the implementation of BFS and DFS for a given graph.
12. Write Java programs for implementing the following sorting methods:

    a)Bubble sort
    b) Insertion sort

1. Write Java programs that use both recursive and non-recursive functions for implementing the Linear Search

PROGRAM: Linear Search using recursive function

```
import java.io.*;
class   RecursiveLinearSearch1
{
        public static int arr[], key;
        public static void main(String args[]) throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); System.out.println("enter n
                value");
                int n=Integer.parseInt(br.readLine());
                arr=new int[n]; System.out.println("enter
                elements"); for(int i=0;i<n;i++)
                {
                        arr[i]=Integer.parseInt(br.readLine());
                }
                System.out.println("enter element to search");
                key=Integer.parseInt(br.readLine());

                if( linearSearch(arr.length-1) )
                        System.out.println(key + " found in the list" );
                else
                         System.out.println(key + " not found in the list");
        }

        static  boolean  linearSearch(int  n)
        {
                if( n < 0 ) return false;
                if(key == arr[n])
                        return true;
                else
                        return  linearSearch(n-1);

        }
}
```

PROGRAM: Linear Search using Non recursive function

```
import java.io.*;
class LinearSearch
{
	public static void main(String args[]) throws IOException
	{
		int count=0;
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); System.out.println("enter n
		value");
		int n=Integer.parseInt(br.readLine());int
		arr[]=new int[n];
		System.out.println("enter elements");
		for(int i=0;i<n;i++)
		{
			arr[i]=Integer.parseInt(br.readLine());
		}
		System.out.println("enter element to search");int
		key=Integer.parseInt(br.readLine()); for(int
		i=0;i<n;i++)
		{
		if(arr[i]==key)
			System.out.println("element found : " + key + " in position :" + (i+1));
		else
			count++;
		}
		if(count==n)
			System.out.println(key + " element not found, search failed");
	}
}
```

2. Write Java programs that use both recursive and non-recursive functions for implementing the Binary Search

PROGRAM: Binary Search using recursive function

```
import java.io.*;
class  RecursiveBinarySearch
{
 public static int arr[], key;
 public static void main(String args[]) throws IOException
 {
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
 System.out.println("enter n value");
 int n=Integer.parseInt(br.readLine());
 arr=new int[n]; System.out.println("enter
 elements"); for(int i=0;i<n;i++)
 {
 arr[i]=Integer.parseInt(br.readLine());
 }
 System.out.println("enter element to  search");
 key=Integer.parseInt(br.readLine());
 if( binarySearch(0, arr.length-1) )
 System.out.println(key + " found in the list");else
 System.out.println(key + " not found in the list");
 }
 static boolean binarySearch(int low, int high)
 {
 if( low > high ) return false;int
 mid = (low + high)/2;
 int c = ((Comparable)key).compareTo(arr[mid]);if( c
 < 0) return binarySearch(low, mid-1);
 else if( c > 0) return binarySearch(mid+1, high);else
 return true;
 }
}
```

PROGRAM: Binary Search using Non recursive function

```
class BinarySearch
{
static Object[] a = { "AP", "KA", "MH", "MP", "OR", "TN", "UP", "WB"};
 static Object key = "UP";
 public static void main(String args[])
 {
 if( binarySearch() )
 System.out.println(key + " found in the list");else
 System.out.println(key + " not found in the list");
 }
 static boolean binarySearch()
 {
 int c, mid, low = 0, high = a.length-1;
 while( low <= high)
 {
 mid = (low + high)/2;
 c = ((Comparable)key).compareTo(a[mid]);if( c
 < 0) high = mid-1;
 else if( c > 0) low = mid+1;
 else return true;
 }
 return false;
 }
}
```

3. Write a Java program to implement the List ADT using arrays

PROGRAM: Implement the List ADT using arrays

```
interface List
 {
public void createList(int n); public
void insertFirst(Object ob);
public void insertAfter(Object ob, Object pos);
public Object deleteFirst();
public Object deleteAfter(Object pos);
public boolean isEmpty();
public int size();
}
class ArrayList implements List
 {
class Node
 {
Object data;
int next;
Node(Object ob, int i) // constructor
{
data = ob;
 next = i;
}
}
int MAXSIZE; // max number of nodes in the list
 Node list[]; // create list array
int head, count; // count: current number of nodes in the listArrayList( int
s) // constructor
{
MAXSIZE = s;
list = new Node[MAXSIZE];
}
public void initializeList()
{
for( int p = 0; p < MAXSIZE-1; p++ )
list[p] = new Node(null, p+1);
list[MAXSIZE-1] = new Node(null, -1);
}
```

```
public void createList(int n) // create ‘n’ nodes
{
int p;
for( p = 0; p < n; p++ )
{
list[p] = new Node(11+11*p, p+1);
count++;
}
list[p-1].next = -1; // end of the list
}
public void insertFirst(Object item)
{
if( count == MAXSIZE )
{
System.out.println("***List is FULL");
 return;
}
int p = getNode();if(
p != -1 )
 {
list[p].data = item;if(
isEmpty() )
list[p].next = -1;
 else
list[p].next = head;
head = p; count++;
}
}
public void insertAfter(Object item, Object x)
{
if( count == MAXSIZE )
 {
System.out.println("***List is FULL");
return;
}
int q = getNode(); // get the available position to insert new nodeint p =
find(x); // get the index (position) of the Object x
if( q != -1 )
{
list[q].data = item; list[q].next =
list[p].next;
```

```
list[p].next = q;
count++;
}
}
public int getNode() // returns available node index
{
for( int p = 0; p < MAXSIZE; p++ )
 if(list[p].data == null)
 return p;
return -1;
}
public int find(Object ob) // find the index (position) of the Object ob
{
int p = head;
 while( p != -1)
 {
if( list[p].data == ob )
 return p;
p = list[p].next; // advance to next node
}
return -1;
}
public Object deleteFirst()
{
if( isEmpty() )
{
System.out.println("List is empty:  no deletion");return
null;
}
Object tmp = list[head].data;
if( list[head].next == -1 ) // if the list contains one node,head = -1;
 // make list empty.
else
head   =   list[head].next;
count--;  //  update  count
return tmp;
}
public Object deleteAfter(Object x)
 {
int p = find(x);
if( p == -1 || list[p].next == -1 )
 {
```

```
System.out.println("No deletion");return
 null;
}
int q = list[p].next; Object
tmp = list[q].data;
list[p].next = list[q].next;
count--;
return tmp;
}
public void display()
 {
 int p = head;
System.out.print("\nList: [ " );
while( p != -1)
 {
System.out.print(list[p].data + "  "); //  print  datap =
list[p].next; // advance to next node
}
System.out.println("]\n");//
}
public boolean isEmpty()
{
if(count == 0)
return true; else
 return false;
}
public int size()
{
return count;
}
}
class  ArrayListDemo  {
public static void main(String[] args)
 {
ArrayList linkedList = new ArrayList(10);
linkedList.initializeList();
linkedList.createList(4);  //  create  4  nodes
linkedList.display(); // print the list
System.out.print("InsertFirst 55:");
linkedList.insertFirst(55); linkedList.display();
```

```
System.out.print("Insert 66 after 33:");
linkedList.insertAfter(66, 33); // insert 66 after 33
linkedList.display();
Object item = linkedList.deleteFirst();
System.out.println("Deleted node: " + item);
linkedList.display();
System.out.print("InsertFirst 77:");
linkedList.insertFirst(77); linkedList.display();
item = linkedList.deleteAfter(22); // delete node after node 22
System.out.println("Deleted node: " + item); linkedList.display();
System.out.println("size(): " + linkedList.size());
}
}
```

4. Write a Java program to implement the List ADT using Linked list

PROGRAM: Implement the List ADT using Linked List

```
class LinkedList implements List
{
class Node
{
Object data; // data item
Node next; // refers to next node in the listNode(
Object d ) // constructor
{
data = d;
} // „next“ is automatically set to null
}
Node head; // head refers to first node
Node p; // p refers to current node
int count; // current number of nodes
public void createList(int n) // create 'n' nodes
{
p = new Node(11); // create first node
head = p; // assign mem. address of 'p' to 'head'for( int i
= 1; i < n; i++ )
{ // create 'n-1' nodes
p = p.next = new Node(11 + 11*i);
}
count = n;
}
public void insertFirst(Object item) // insert at the beginning of list
{
p = new Node(item); // create new node p.next =
head; // new node refers to old headhead = p; //
new head refers to new node count++;
}
public void insertAfter(Object item,Object key)
{
 p = find(key); // get “location of key item”if( p
== null )
```

```
System.out.println(key + " key is not found");else
{
Node q = new Node(item); // create new node q.next =
p.next; // new node next refers to p.nextp.next = q; //
p.next refers to new node count++;
}
}
public Node find(Object key)
{
p = head;
while( p != null ) // start at beginning of list until end of list
{
if( p.data == key )
 return p;
p = p.next; // move to next node
}
return null; // if key search is unsuccessful,
}
public Object deleteFirst()
{ // delete first nodeif(
isEmpty() )
{
System.out.println("List is empty:  no deletion");return
null;
}
Node tmp = head; // tmp saves reference to headhead =
 tmp.next;
count--;
return tmp.data;
}

public Object deleteAfter(Object key) // delete node after key item
{
p = find(key); // p = “location of key node”if( p
== null )
{
System.out.println(key + " key is not found");return
 null;
}
if( p.next == null ) // if(there is no node after key node)
```

```
{
System.out.println("No deletion");return
null;
}
else
{
Node tmp = p.next; // save node after key node p.next =
tmp.next; // point to next of node deletedcount--;
return tmp.data; // return deleted node
}
}
public void displayList()
 {
p = head; // assign mem. address of 'head' to 'p'
System.out.print("\nLinked List: ");
while( p != null ) // start at beginning of list until end of list
{
System.out.print(p.data + " -> "); // print datap =
p.next; // move to next node
}
System.out.println(p);  //  prints  'null'
}
public boolean isEmpty() // true if list is empty
{
return (head == null);
}
public int size()
{
return count;
}
} // end of LinkedList class
class LinkedListDemo
{
public static void main(String[] args)
{
LinkedList list = new LinkedList(); // create list object
list.createList(4);// create 4 nodes
list.displayList();
list.insertFirst(55);  //  insert  55  as  first  node
list.displayList();
list.insertAfter(66, 33); // insert 66 after 33
```

```
list.displayList();
Object item = list.deleteFirst(); // delete first nodeif(
item != null )
{
System.out.println("deleteFirst(): "  +  item);
list.displayList();
}
item = list.deleteAfter(22); // delete a node after node(22)if( item
!= null )
{
System.out.println("deleteAfter(22): " + item);
 list.displayList();
}
System.out.println("size():  "  +  list.size());
}
}
```

5. Write Java program to implement the stack ADT using array

PROGRAM: Implement the Stack ADT using Array

```
import java.io.*;
class stackclass
{
int top,ele,stack[],size;
stackclass(int n)
{
stack=new int[n];
size=n;
top= -1;
}
void push(int x)
{
ele=x;
stack[++top]=ele;
}
int pop()
{
if(!isempty())
{
 System.out.println("Deleted element is");return
stack[top--];
}
else
{
System.out.println("stack is empty");return
-1;
 }
}
boolean isempty()
{
if(top==-1)
return true;
```

```
else
return false;
}
boolean isfull()
{
if(size>(top+1))
return false; else
return true;
}
int peek()
{
if(!isempty()) return
stack[top];else
{
System.out.println("stack is empty");return
-1;
}
}
void size()
{
System.out.println("size of the stack is :"+(top+1));
}
void display()
{
if(!isempty())
{
for(int i=top;i>=0;i--)
System.out.print(stack[i]+" ");
}
else
System.out.println("stack is empty");
 }
}
class stacktest
{
public static void main(String args[])throws Exception
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter the size of stack");
int size=Integer.parseInt(br.readLine());
```

```
stackclass s=new  stackclass(size);int
ch,ele;
do
{
System.out.println();
System.out.println("1.push");
System.out.println("2.pop");
System.out.println("3.peek");
System.out.println("4.size");
System.out.println("5.display");
System.out.println("6.is empty");
System.out.println("7.is full");
System.out.println("8.exit");
System.out.println("enter ur choise :");
ch=Integer.parseInt(br.readLine());
switch(ch)
{
case  1:if(!s.isfull())
{
 System.out.println("enter the element to insert: ");
ele=Integer.parseInt(br.readLine());
s.push(ele);
}
else
{
System.out.print("stack  is  overflow");
}
break;
case 2:int del=s.pop();
if(del!=-1)
System.out.println(del+" is deleted");
break;
case  3:int  p=s.peek();
if(p!=-1)
System.out.println("peek element  is:" +p);break;
case 4:s.size();break;
case 5:s.display();
break;
case 6:boolean b=s.isempty();System.out.println(b);
```

```
break;
case 7:boolean b1=s.isfull();
System.out.println(b1); break;
case 8 :System.exit(1);
}
}while(ch!=0);
}
}
```

6. Write Java program to implement the QueueADT using array

PROGRAM: Implement the Queue ADT using Array

```
import java.util.*;
class queue
{
 int front,rear;
 int que[];
 int max,count=0;
 queue(int n)
 {
 max=n;
 que=new int[max];
 front=rear=-1;
 }
 boolean isfull()
 {
 if(rear==(max-1))
 return true;
 else
 return false;
 }
 boolean isempty()
 {
 if(front==-1)
 return true;
 else
 return false;
 }
 void insert(int n)
 {
 if(isfull()) System.out.println("list
 is full");else
 {
 rear++;
 que[rear]=n;
 if(front==-1)
```

```
 front=0;
count++;
 }
 }
 int delete()
 {
 int x;
 if(isempty())
 return -1;
 else
 {
 x=que[front];
 que[front]=0;
 if(front==rear)
 front=rear=-1;
 else
 front++;
 count--;
 }
 return x;
 }
 void display()
 {
 if(isempty()) System.out.println("queue
 is empty");else
 for(int i=front;i<=rear;i++)
 System.out.println(que[i]);
 }
 int size()
 {
 return count;
 }
 public static void main(String args[])
 {
 int ch;
 Scanner s=new Scanner(System.in);
 System.out.println("enter limit"); int
 n=s.nextInt();
 queue q=new queue(n);do
 {
 System.out.println("1.insert");
 System.out.println("2.delete");
 System.out.println("3.display");
 System.out.println("4.size");
```

```
 System.out.println("enter ur choise :");
 ch=s.nextInt();
 switch(ch)
 {
 case 1:System.out.println("enter element :");int
 n1=s.nextInt();
 q.insert(n1);
 break;
case 2:int c1=q.delete();
 if(c1>0)
 System.out.println("deleted element is :"+c1);else
 System.out.println("can't delete");
 break;
 case 3:q.display();
 break;
 case 4:System.out.println("queue size is "+q.size());break;
 }
 }
 while(ch!=0);
 }
}
```

7. Write a java program that reads an infix expression, converts the expression to postfix form and then evaluates the postfix expression (use stack ADT).

PROGRAM: Infix to Postfix Conversion

```
import java.io.*;
class InfixToPostfix
{
java.util.Stack<Character> stk =new java.util.Stack<Character>();public String
toPostfix(String infix)
{
infix = "(" + infix + ")"; // enclose infix expr within parenthesesString
postfix = "";
/* scan the infix char-by-char until end of string is reached */for( int
i=0; i<infix.length(); i++)
{
char ch, item;
ch = infix.charAt(i);
if( isOperand(ch) ) // if(ch is an operand), then postfix =
postfix + ch; // append ch to postfix stringif( ch == '(' ) //
if(ch is a left-bracket), then stk.push(ch); // push onto the
stack
if( isOperator(ch) ) // if(ch is an operator), then
{
item = stk.pop(); // pop an item from the stack
/* if(item is an operator), then check the precedence of ch and item*/if(
isOperator(item) )
{
if( precedence(item) >= precedence(ch) )
{
stk.push(item);
stk.push(ch);
}
else
{
postfix = postfix + item;
stk.push(ch);
}
}
else
{
stk.push(item);
stk.push(ch);
```

```
}
} // end of if(isOperator(ch))if(
ch == ')' )
{
item = stk.pop();
while( item != '(' )
{
postfix = postfix + item;
item = stk.pop();
}
}
} // end of for-loop
return postfix;
} // end of toPostfix() method
public boolean isOperand(char c)
{
return(c >= 'A' && c <= 'Z');
}
public boolean isOperator(char c)
{
return( c=='+' || c=='-' || c=='*' || c=='/' );
}
public int precedence(char c)
{
int rank = 1; // rank = 1 for '*' or '/'if( c
== '+' || c == '-' ) rank = 2; return rank;
}
}
//InfixToPostfixDemo.java
class InfixToPostfixDemo
{
public static void main(String args[]) throws IOException
{
InfixToPostfix obj = new InfixToPostfix();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter Expression:");
String infix = br.readLine();
//String infix = "A*(B+C/D)-E"; System.out.println("infix:
" + infix ); System.out.println("postfix:"+obj.toPostfix(infix)
);
}
}
```

8. Write a Java program to implement the Stack ADT using a singly linked list.

PROGRAM: Stack ADT using Single Linked list

```
 import java.io.*;
class Stack1
{
Stack1 top,next,prev;
int data;
Stack1()
{
data=0;
next=prev=null;
}
Stack1(int d)
{
data=d;
next=prev=null;
}
void push(int n)
{
Stack1 nn; nn=new
Stack1(n);
if(top==null)
top=nn;
else
{
nn.next=top;
top.prev=nn;
top=nn;
}
}
int pop()
{
int k=top.data;
if(top.next==null)
{
top=null;
return k;
}
```

```
else
{
top=top.next;
top.prev=null;
return k;
}
}
boolean isEmpty()
{
if (top==null)
return true;
else
return false;
}
void display()
{
Stack1 ptr;
for(ptr=top;ptr!=null;ptr=ptr.next)
System.out.print(ptr.data+" ");
}
public static void main(String args[ ])throws Exception
{
int x;
int ch;
BufferedReader b=new BufferedReader(new InputStreamReader(System.in));Stack1
a=new Stack1();
do{
System.out.println("enter 1 for pushing");
System.out.println("enter 2 for poping");
System.out.println("enter 3 for isEmpty");
System.out.println("enter 4 for display");
System.out.println("Enter 0 for exit");
System.out.println("enter ur choice ");
ch=Integer.parseInt(b.readLine());
switch(ch)
{
case 1:System.out.println("enter element to insert");int
e=Integer.parseInt(b.readLine());
a.push(e);
break;
case 2:if(!a.isEmpty())
{
int p=a.pop();
System.out.println("deleted element is "+p);
}
else
```

```
{
System.out.println("stack is empty");
}
break;
case 3:System.out.println(a.isEmpty());
break;
case 4:if(!a.isEmpty())
{
a.display();
}
else
{
System.out.println("list is empty");
}
}
}
while(ch!=0);
}
}
```

9. Write a Java program to implement the Queue ADT using a singly linked list.

PROGRAM: Queue ADT using Single Linked list

```
import java.io.*;
class Qlnk
{
Qlnk front,rear,next;
int data;
Qlnk()
{
data=0;
next=null;
}
Qlnk(int d)
{
data=d;
next=null;
}
Qlnk getFront()
{
return front;
}
Qlnk getRear()
{
return rear;
}
void insertelm(int item)
{
Qlnk nn;
nn=new Qlnk(item);
if(isEmpty())
{
front=rear=nn;
}
else
```

```
{
rear.next=nn;
rear=nn;
}
}
int delelm()
{
if(isEmpty())
{
System.out.println("deletion failed");return
-1;
}
else
{
int k=front.data;
if(front!=rear)
front=front.next;else
rear=front=null;
return k;
}
}
boolean isEmpty()
{
if(rear==null)
return true;
else
return false;
}
int size()
{
Qlnk ptr;
int cnt=0;
for(ptr=front;ptr!=null;ptr=ptr.next)cnt++;
return cnt;
}
void display()
{
Qlnk ptr;
if(!isEmpty())
{
```

```
for(ptr=front;ptr!=null;ptr=ptr.next)
System.out.print(ptr.data+" ");
}
else
System.out.println("q  is  empty");
}
public static void main(String arr[])throws Exception
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));Qlnk
m=new Qlnk();
int ch;
do
{
System.out.println("enter 1 for insert");
System.out.println("enter 2 for deletion");
System.out.println("enter 3 for getFront");
System.out.println("enter 4 for getRear");
System.out.println("enter 5 for size");
System.out.println("enter 6 for display");
System.out.println("enter 0 for exit");
System.out.println("enter ur choice");
ch=Integer.parseInt(br.readLine()); switch(ch)
{
case 1:System.out.println("enter ele to insert");int
item=Integer.parseInt(br.readLine());
m.insertelm(item);break;
case 2:int k=m.delelm(); System.out.println("deleted ele
is  "+k);break;
case 3:System.out.println("front index is"+(m.getFront()).data);break; case
4:System.out.println("rear  index is"+(m.getRear()).data);break; case
5:System.out.println("size is"+m.size());break;
case  6:m.display();break;
}
}while(ch!=0);
}
}
```

10. Write Java programs that use recursive and non-recursive functions to traverse the given binarytree in a) Preorder b) Inorder c) Postorder.

PROGRAM: Traverse the given binary tree in a) Preorder b) Inorder c) Postorder.

```
class Node
{
 Object data;
 Node left;
 Node right;
 Node( Object d ) // constructor
 {
 data = d;
 }
}
class BinaryTree
{
 Object tree[];
 int maxSize;
 java.util.Stack<Node> stk = new java.util.Stack<Node>();
 BinaryTree( Object a[], int n ) // constructor
 {
 maxSize = n;
 tree = new Object[maxSize];
 for( int i=0; i<maxSize; i++ )
 tree[i] = a[i];
 }
 public Node buildTree( int index )
 {
 Node p = null;
 if( tree[index] != null )
 {
 p = new Node(tree[index]); p.left
 = buildTree(2*index+1); p.right =
 buildTree(2*index+2);
 }
 return p;
 }
 /* Recursive methods - Binary tree traversals */public
 void inorder(Node p)
```

```
{
if( p != null )
{
inorder(p.left);
System.out.print(p.data + " ");
inorder(p.right);
}
}
public void preorder(Node p)
{
if( p != null )
{
System.out.print(p.data + " ");
preorder(p.left);
preorder(p.right);
}
}
public void postorder(Node p)
{
if( p != null )
{
postorder(p.left);
postorder(p.right);
System.out.print(p.data + " ");
}
}
/* Non-recursive methods - Binary tree traversals */public
void preorderIterative(Node p)
{
if(p == null )
{
System.out.println("Tree is empty");
return;
}
stk.push(p);
while( !stk.isEmpty() )
{
p = stk.pop();if(
p != null )
{
System.out.print(p.data + " ");
stk.push(p.right);
stk.push(p.left);
}
}
}
```

```
public void inorderIterative(Node p)
{
if(p == null )
{
System.out.println("Tree is empty");
return;
}
while( !stk.isEmpty() || p != null )
{
if( p != null )
{
stk.push(p); // push left-most path onto stackp =
p.left;
}
else
{
p = stk.pop(); // assign popped node to p
System.out.print(p.data + " "); // print node datap =
p.right; // move p to right subtree
}
}
}
public void postorderIterative(Node p)
{
if(p == null )
{
System.out.println("Tree is empty");
return;
}
Node tmp = p;
while( p != null )
{
while( p.left != null )
{
stk.push(p);p
= p.left;
}
while( p != null && (p.right == null || p.right == tmp ))
{
System.out.print(p.data + " "); // print node datatmp = p;
if( stk.isEmpty() )
return;
p = stk.pop();
}
stk.push(p);
```

```
 p = p.right;
 }
 }
} // end of BinaryTree class
class BinaryTreeDemo
{
 public static void main(String args[])
 {
 Object arr[] = {'E', 'C', 'G', 'A', 'D', 'F', 'H', null,'B',
 null, null, null, null, null, null, null, null, null, null };
 BinaryTree t = new BinaryTree( arr, arr.length );
 Node root = t.buildTree(0); // buildTree() returns reference to root
 System.out.print("\n Recursive Binary Tree Traversals:");
 System.out.print("\n inorder: ");
 t.inorder(root); System.out.print("\n
 preorder: "); t.preorder(root);
 System.out.print("\n postorder: ");
 t.postorder(root);
 System.out.print("\n Non-recursive Binary Tree Traversals:");
 t.inorderIterative(root);
 System.out.print("\n preorder: ");
 t.preorderIterative(root);
 System.out.print("\n postorder: ");
 t.postorderIterative(root);
 }
}
```

11. Write Java programs for the implementation of BFS and DFS for a given graph.

PROGRAM: Implementation of BFS a given graph.

```
import java.io.*;
class quelist
{
public int front;
public int rear;
public int maxsize;
public int[] que;
public quelist(int size)
{
maxsize = size;
que = new int[size];
front = rear = -1;
}
public void display()
{
for(int i = front;i<=rear;i++)
System.out.print(que[i]+" ");
}
public void enque(int x)
{
if(front==-1)
front = 0;
que[++rear]=x;
}
public int deque()
{
int temp = que[front];
front = front +1; return
temp;
}
public boolean isempty()
{
return((front>rear)||(front==-1));
}
}
class vertex
```

```
{
public char label;
public boolean wasvisited;
public vertex(char lab)
{
label = lab;
wasvisited = false;
}
}
class graph
{
public final int MAX = 20;
public int nverts;
public int adj[][];
public vertex vlist[];
quelist qu;
public graph()
{
nverts = 0;
vlist = new vertex[MAX];
adj = new int[MAX][MAX];
qu = new quelist(MAX);
for(int i=0;i<MAX;i++)
for(int j=0;j<MAX;j++)
adj[i][j] = 0;
}
public void addver(char lab)
{
vlist[nverts++] = new vertex(lab);
}
public void addedge(int start,int end)
{
adj[start][end] = 1;
adj[end][start] = 1;
}
public int getadjunvis(int i)
{
for(int j=0;j<nverts;j++)
if((adj[i][j]==1)&&(vlist[j].wasvisited==false))return j;
return (MAX+1);
}
public void display(int i)
{
System.out.print(vlist[i].label);
}
```

```
public int getind(char l)
{
for(int i=0;i<nverts;i++)
if(vlist[i].label==l) return i;
return (MAX+1);
}
public void brfs()
{
vlist[0].wasvisited = true;
display(0);
qu.enque(0);
int v2;
while(!(qu.isempty()))
{
int v1 = qu.deque();
while((v2=getadjunvis(v1))!=(MAX+1))
{
vlist[v2].wasvisited = true;
display(v2); qu.enque(v2);
}
}
System.out.print("\n");
}
}
class bfs
{
public static void main(String args[])throws IOException
{
graph gr = new graph();
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
System.out.println("Enter the number of vertices");
int n = Integer.parseInt(br.readLine());
System.out.println("Enter the labels for the vertices");for(int
i=0;i<n;i++)
{
String temp = br.readLine();
char ch = temp.charAt(0);
gr.addver(ch);
}
System.out.println("Enter the number of edges");int
edg = Integer.parseInt(br.readLine());
System.out.println("Enter the vertices which you need to connect");for(int
j=0;j<edg;j++)
```

```
{
System.out.println("Enter the first vertex");
String t = br.readLine();
char c = t.charAt(0); int
start = gr.getind(c);
System.out.println("Enter the second vertex");t =
br.readLine();
c = t.charAt(0);
int end = gr.getind(c);
gr.addedge(start,end);
}
System.out.print("The vertices in the graph traversed breadthwise:");gr.brfs();
}
}
```

PROGRAM: Implementation of DFS a given graph.

```
import java.io.*;
import java.util.*;
class Stack
{
 int stk[]=new int[10];int
 top;
 Stack()
 {
 top=-1;
 }
 void push (int item)
 {
 if (top==9) System.out.println("Stack
 overflow");else
 stk[++top]=item;
 }/*end push*/
 boolean isempty()
 {
 if (top<0)
 return true;
 else
 return false;
 }/*end isempty*/
 int pop()
 {
 if (isempty())
```

```
 {
 System.out.println("Stack underflow");
 return 0;
 }
 else
 return (stk[top--]);
 }/*end pop*/ void
 stacktop()
 {
 if(isempty())
 System.out.println("Stack underflow ");else
 System.out.println("Stack top is "+(stk[top]));
 }/*end stacktop*/
 void display()
 {
 System.out.println("Stack-->");
 for(int i=0;i<=top;i++)
 System.out.println(stk[i]);
 }/*end display*/
}
class Graph
{
 int MAXSIZE=51;
 int adj[][]=new int[MAXSIZE][MAXSIZE];
 int visited[]=new int [MAXSIZE];
 Stack s=new Stack();
 /*Function for Depth-First-Search */void
 createGraph()
 {
 int n,i,j,parent,adj_parent,initial_node;int
 ans=0,ans1=0;
 System.out.print("\nEnter total number elements in a Undirected Graph :");n=getNumber();
 for   (   i=1;i<=n;i++)
 for(      j=1;j<=n;j++)
 adj[i][j]=0;
 /*All graph nodes are unvisited, hence assigned zero to visited field of each node */for (int
 c=1;c<=50;c++)
 visited[c]=0;
 System.out.println("\nEnter graph structure for BFS ");do
 {
 System.out.print("\nEnter parent node :");
 parent=getNumber();
 do
```

```
{
System.out.print("\nEnter adjacent node for node "+parent+ " : ");adj_parent=getNumber();
adj[parent][adj_parent]=1;
adj[adj_parent][parent]=1;
System.out.print("\nContinue to add adjacent node for "+parent+"(1/0)?");ans1=
getNumber();
} while (ans1==1);
System.out.print("\nContinue to add graph node?");ans=
getNumber();
}while (ans ==1);
System.out.print("\nAdjacency matrix for your graph is :\n");for
(i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
System.out.print(" "+adj[i][j]);
System.out.print("\n");
}
System.out.println("\nYour Undirected Graph is :");for
(i=1;i<=n;i++)
{
System.out.print("\nVertex "+i+"is connected to : ");for
(j=1;j<=n;j++)
{
if (adj[i][j]==1)
System.out.print(" "+j);
}
}
System.out.println("\nEnter the initial node for BFS traversal:");
initial_node=getNumber();
DFS (initial_node, n);
}
void DFS (int initial_node,int n)
{
int u,i;
s.top = -1;
s.push(initial_node);
System.out.println("\nDFS traversal for given graph is : ");
while(!s.isempty())
{
u=s.pop();
if(visited[u]==0)
{
System.out.print("\n"+u);
visited[u]=1;
}
```

```
 for (i=1;i<=n;i++)
 {
 if((adj[u][i]==1) && (visited[i]==0))
 {
 s.push(u); visited[i]=1;
 System.out.print(" "+i);u
 = i;
 }
 }
 }
 }/* end of DFS function */int
 getNumber()
 {
 String str;
 int ne=0;
 InputStreamReader input=new InputStreamReader(System.in);
 BufferedReader in=new BufferedReader(input);
 try
 {
 str=in.readLine(); ne=Integer.parseInt(str);
 }
 catch(Exception e)
 {
 System.out.println("I/O Error");
 }
 return ne;
 }
}
class Graph_DFS
{
 public static void main(String args[])
 {
 Graph g=new Graph();
 g.createGraph();
 } /* end of program */
}
```

12. Write Java programs for implementing the following sorting methods:
a) Bubble sort b) Insertion sort

PROGRAM: Bubble sort

```
import java.io.*;
class BubbleSort
{
public static void main(String[] args) throws IOException
{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("enter n value");
int n=Integer.parseInt(br.readLine());int
arr[]=new int[n];
System.out.println("enter elements");
for(int i=0;i<n;i++)
{
arr[i]=Integer.parseInt(br.readLine());
}
System.out.print("\n Unsorted array: ");display(
arr );
bubbleSort( arr ); System.out.print("\n
Sorted array: ");display( arr );
}
static void bubbleSort(int[] a)
{
int i, pass, exch, n = a.length;int
tmp;
for( pass = 0; pass < n; pass++ )
{
exch = 0;
for( i = 0; i < n-pass-1; i++ )
if( ((Comparable)a[i]).compareTo(a[i+1]) > 0)
{
tmp = a[i]; a[i]
= a[i+1];a[i+1]
= tmp;exch++;
}
```

```
if( exch == 0 ) return;
}
}
static void display( int a[] )
{
for( int i = 0; i < a.length; i++ )
System.out.print( a[i] + " " );
}
}
```

PROGRAM: Insertion sort

```
import java.io.*;
class InsertionSort
{
public static void main(String[] args) throws IOException
{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("enter n value");
int n=Integer.parseInt(br.readLine());int
arr[]=new int[n];
System.out.println("enter elements");
for(int i=0;i<n;i++)
{
arr[i]=Integer.parseInt(br.readLine());
}
System.out.print("\n Unsorted array: ");
display( arr );
insertionSort( arr ); System.out.print("\n
Sorted array: ");display( arr );
}
static void insertionSort(int a[])
{
int i, j, n = a.length;int
item;
for( j = 1; j < n; j++ )
{
item = a[j];i
= j-1;
while( i >= 0 && ((Comparable)item).compareTo(a[i]) < 0)
{
a[i+1] = a[i];i
= i-1;
```

```
}
a[i+1] = item;
}
}
static void display( int a[] )
{
for( int i = 0; i < a.length; i++ )
System.out.print( a[i] + " " );
}
}
```