ASSIGNMENT 2

**Name: Soham Makarand Limaye**

**Roll No: 222038**

**PRN: 22111058**

**Batch: B2**

**Aim:** Construct an expression tree from postfix/prefix expression and perform recursive and non\_recursive In-order, pre-order and post-order traversals.

**Objective :**

1. To understand the implementation of the tree using java
2. to perform the various operation on tree data structure

3)to understand oops in java 4)to understand the binary search tree

**Theory :**

In this program first we take the postfix expression as a input and

and we converted that expression into tree form and using stack we display it in preorder, post order ,inorder with recursion and without recursion .

**Code :**

import java.util.\*;

class TBSTNode

{ int ele;

TBSTNode left, right; boolean leftThread, rightThread; public

TBSTNode(int ele)

{ this(ele, null, null, true, true);

}

public TBSTNode(boolean leftThread, boolean rightThread)

{

this.ele = Integer.MAX\_VALUE;

this.left = this; this.right = this; this.leftThread = leftThread; this.rightThread

= rightThread;

}

public TBSTNode(int ele, TBSTNode left, TBSTNode right, boolean

leftThread, boolean rightThread)

{ this.ele = ele; this.left = left; this.right = right; this.leftThread = leftThread; this.rightThread = rightThread;

}

}

class ThreadedBinarySearchTree

{

private TBSTNode root; public

ThreadedBinarySearchTree ()

{

root = new TBSTNode(true, false);

}

public void clear()

{

root = new TBSTNode(true, false);

}

public void insert(int ele)

{

TBSTNode ptr = findNode(root,

ele); if (ptr == null) return; if (ptr.ele

< ele)

{

TBSTNode nptr = new TBSTNode(ele, ptr, ptr.right, true, true);

ptr.right = nptr; ptr.rightThread = false;

}

else

{

TBSTNode nptr = new TBSTNode(ele, ptr.left, ptr, true, true);

ptr.left = nptr;

ptr.leftThread = false;

}

}

public TBSTNode findNode(TBSTNode r, int ele)

{

if (r.ele < ele)

{ if (r.rightThread) return r; return findNode(r.right, ele);

} else if (r.ele > ele)

{ if (r.leftThread) return r; return findNode(r.left, ele);

}

else return

null;

}

public void inOrder()

{

TBSTNode temp = root;

for (;;) {

temp = insucc(temp); if (temp == root) break;

System.out.print(temp.ele +" ");

}

}

public TBSTNode insucc(TBSTNode tree)

{

TBSTNode temp;

temp = tree.right; if

(!tree.rightThread) while (!temp.leftThread) temp = temp.left; return temp;

}

}

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

{

Scanner scan = new Scanner(System.in);

ThreadedBinarySearchTree tbst = new

ThreadedBinarySearchTree();

System.out.println("Threaded Binary Search Tree Test\n"); char ch; do

{

System.out.println("1. insert "); System.out.println("2. clear"); int choice = scan.nextInt(); switch (choice)

{

case 1 :

System.out.println("Enter integer element to insert"); tbst.insert( scan.nextInt() ); break; case 2 :

System.out.println("\nTree Cleared\n"); tbst.clear(); break; default :

System.out.println("Wrong Entry \n "); break;

}

System.out.print("\nTree = "); tbst.inOrder();

System.out.println();

System.out.println("\nDo you want to continue (Type y or n)

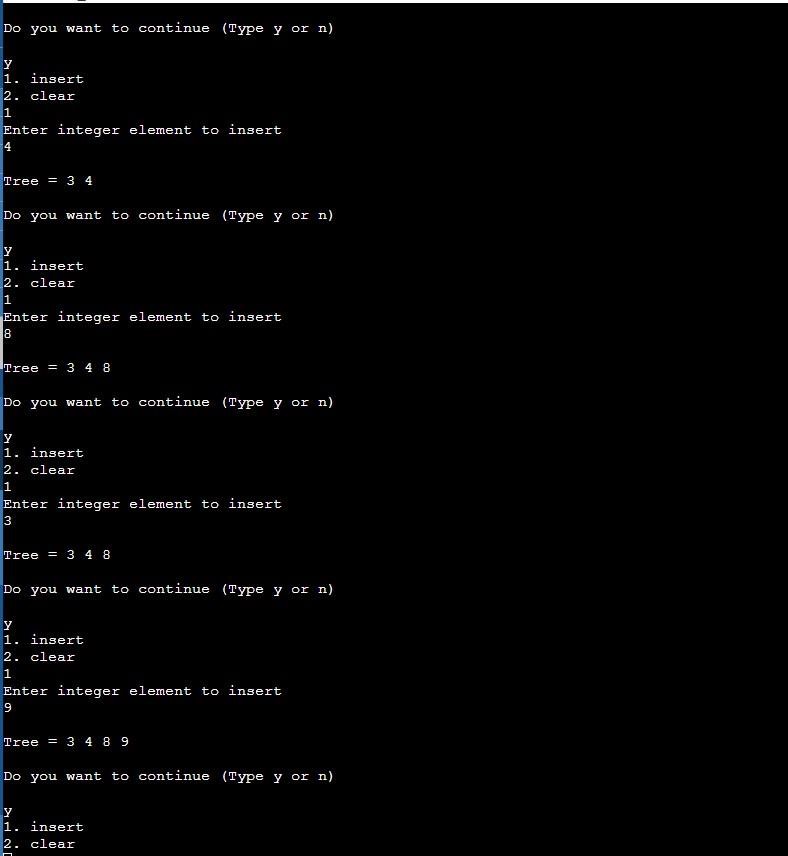
\n");

ch = scan.next().charAt(0);

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

} }

**Output :**



**Conclusion:**

In this program we converted the postfix expression into tree and after we display it using the preorder ,postorder,inorder with recursion and without recursion.