

# Postorder Traversal 1

Problem

Submissions

Leaderboard

Discussions

Write a java program to perform postorder tree traversal.

## Input Format

1 10 1 20 1 30 1 40 1 60 1 50 2 3

## Constraints

No Constraints

## Output Format

Postorder Traversal is: 50 60 40 30 20 10

## Sample Input 0

```
1
10
1
20
1
30
1
40
1
60
1
50
2
3
```

## Sample Output 0

```
Postorder Traversal is:
50 60 40 30 20 10
```

[f](#) [t](#) [in](#)**Contest ends in 2 months**

Submissions: 98

Max Score: 10

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

java 7



```
1 //224G1A0553
2 import java.util.*;
3 class Node {
4     int data;
5     Node left;
6     Node right;
7     public Node( int item) {
8         this.data = item;
9         this.left = null;
```

```
10     this.right = null;
11 }
12 class StackNode {
13     Node node;
14     StackNode next;
15     public void StackNode(Node b) {
16         this.node = b;
17         this.next = null;
18     }
19     public class NonRecursivePostorder {
20         StackNode top;
21         Node root;
22         public void NonRecursivePostorder() {
23             top = null;
24             root = null;
25         }
26         boolean isEmpty() {
27             if(top == null) {
28                 return true;
29             }
30             return false;
31         }
32         void push(Node b) {
33             StackNode temp;
34             temp = new StackNode();
35             if(temp == null) {
36                 System.out.printf("Stack is overflow.\n");
37             } else {
38                 temp.node = b;
39                 temp.next = top;
40                 top = temp;
41             }
42         }
43         Node peek() {
44             if (top == null) {
45                 return null;
46             }
47             return top.node;
48         }
49         Node pop() {
50             StackNode temp;
51             Node b;
52             if(top == null) {
53                 System.out.printf("Stack is underflow.\n");
54                 return null;
55             } else {
56                 temp = top;
57                 top = top.next;
58                 b = temp.node;
59                 return b;
60             }
61         }
62         void postorderInBST(Node root) {
63             do {
64                 while(root != null) {
65                     if(root.right != null) {
66                         push(root.right);
67                     }
68                     push(root);
69                     root = root.left;
70                 }
71                 root = pop();
72                 if(root.right != null && peek() == root.right) {
73                     pop();
74                     push(root);
75                     root = root.right;
76                 } else {
77                     System.out.printf("%d ",root.data);
```

```
76         root = null;
77     }
78     } while(!isEmpty());
79 }
80 /* Insertion into binary search tree */
81 Node insertBinarySearchTree(Node root, int item) {
82
83     /* If the tree is empty new node became root */
84     if (root == null) {
85         root = new Node(item);
86         return root;
87     }
88     /* Otherwise, if item is less then root then recur left side */
89     if (item < root.data)
90         root.left = insertBinarySearchTree(root.left, item);
91     else if (item > root.data)
92         root.right = insertBinarySearchTree(root.right, item);
93
94     /* return the root node pointer */
95     return root;
96 }
97 // Driver main method Code
98 public static void main(String[] args) {
99     NonRecursivePostorder tree = new NonRecursivePostorder();
100     Scanner sc = new Scanner(System.in);
101     int option;
102     int item;
103     //System.out.println("Enter 1 to insert\nEnter 2 to display BST in postorder\nEnter 3 to
Exit");
104     while(true) {
105         //System.out.print("Enter your option: ");
106         option = sc.nextInt();
107         switch(option) {
108             default:
109                 System.out.println("Enter the right option");
110                 break;
111             case 1:
112                 //System.out.print("Enter the element to insert: ");
113                 item = sc.nextInt();
114                 tree.root = tree.insertBinarySearchTree(tree.root, item);
115                 break;
116             case 2:
117                 if(tree.root == null) {
118                     System.out.println("Tree is empty, root is null");
119                 }else {
120                     System.out.println("Postorder Traversal is:");
121                     tree.postorderInBST(tree.root);
122                     System.out.println();
123                 }
124                 break;
125             case 3:
126                 return;
127         }
128     }
129 }
```

Line: 11 Col: 6

[Upload Code as File](#) ☐ Test against custom input

Run Code

Submit Code

Testcase 0 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

**Input (stdin)**

```
1
10
1
20
1
30
1
40
1
60
1
50
2
3
```

**Your Output (stdout)**

```
Postorder Traversal is:
50 60 40 30 20 10
```

**Expected Output**

```
Postorder Traversal is:
50 60 40 30 20 10
```