

# 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 9 days

Submissions: [111](#)

Max Score: 10

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

Java 7

```

1 import java.util.*;
2 class Node {
3     int data;
4     Node left;
5     Node right;
6     public Node( int item) {
7         this.data = item;
8         this.left = null;
9         this.right = null;
10    }
11 }
12
13 class StackNode {
14     Node node;
15     StackNode next;
16     public void StackNode(Node b) {
    
```

```

18         this.node = b;
19         this.next = null;
20     }
21 }
22 public class NonRecursivePostorder {
23     StackNode top;
24     Node root;
25     public void NonRecursivePostorder() {
26         top = null;
27         root = null;
28     }
29     boolean isEmpty() {
30         if(top == null) {
31             return true;
32         }
33         return false;
34     }
35     void push(Node b) {
36         StackNode temp;
37         temp = new StackNode();
38         if(temp == null) {
39             System.out.printf("Stack is overflow.\n");
40         } else {
41             temp.node = b;
42             temp.next = top;
43             top = temp;
44         }
45     }
46     Node peek() {
47         if (top == null) {
48             return null;
49         }
50         return top.node;
51     }
52     Node pop() {
53         StackNode temp;
54         Node b;
55         if(top == null) {
56             System.out.printf("Stack is underflow.\n");
57             return null;
58         } else {
59             temp = top;
60             top = top.next;
61             b = temp.node;
62             return b;
63         }
64     }
65     void postorderInBST(Node root) {
66         do {
67             while(root != null) {
68                 if(root.right != null) {
69                     push(root.right);
70                 }
71                 push(root);
72                 root = root.left;
73             }
74             root = pop();
75             if(root.right != null && peek() == root.right) {
76                 pop();
77                 push(root);
78                 root = root.right;
79             } else {
80                 System.out.printf("%d ",root.data);
81                 root = null;
82             }
83         } while(!isEmpty());
84     }
85     /* Insertion into binary search tree */
86     Node insertBinarySearchTree(Node root, int item) {
87
88         /* If the tree is empty new node became root */
89         if (root == null) {

```

```

    root = new Node(item);
    return root;
}

/* Otherwise, if item is less than root then recur left side */
if (item < root.data)
    root.left = insertBinarySearchTree(root.left, item);
else if (item > root.data)
    root.right = insertBinarySearchTree(root.right, item);

/* return the root node pointer */
return root;
}

// Driver main method Code
public static void main(String[] args) {
    NonRecursivePostorder tree = new NonRecursivePostorder();
    Scanner sc = new Scanner(System.in);
    int option;
    int item;
    //System.out.println("Enter 1 to insert\nEnter 2 to display BST in postorder\nEnter 3 to
Exit");
    while(true) {
        //System.out.print("Enter your option: ");
        option = sc.nextInt();
        switch(option) {
            default:
                System.out.println("Enter the right option");
                break;
            case 1:
                //System.out.print("Enter the element to insert: ");
                item = sc.nextInt();
                tree.root = tree.insertBinarySearchTree(tree.root, item);
                break;
            case 2:
                if(tree.root == null) {
                    System.out.println("Tree is empty, root is null");
                }else {
                    System.out.println("Postorder Traversal is:");
                    tree.postorderInBST(tree.root);
                    System.out.println();
                }
                break;
            case 3:
                return;
        }
    }
}
}

```

Line: 1 Col: 1

 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
```