**Preorder to PostOrder**

**Medium**

Given an array arr[] of N nodes representing preorder traversal of some BST. You have to build the exact PostOrder from it's given preorder traversal.   
In Pre-Order traversal, **the root node is visited before the left child and right child nodes**.

**Example 1:**

**Input:**

N = 5

arr[] = {40,30,35,80,100}

**Output:** 35 30 100 80 40

**Explanation:** PreOrder: 40 30 35 80 100

Therefore, the BST will be:

              40

           /      \

         30       80

           \        \

           35      100

Hence, the postOrder traversal will

be: 35 30 100 80 40

**Example 2:**

**Input:**

N = 8

arr[] = {40,30,32,35,80,90,100,120}

**Output:** 35 32 30 120 100 90 80 40

**Expected Time Complexity:**O(N).  
**Expected Auxiliary Space:**O(N).

**Constraints:**  
1 <= N <= 103  
1 <= arr[i] <= 104

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon)[**L inkedin**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Linkedin)

//{ Driver Code Starts

import java.util.\*;

import java.io.\*;

class Node {

int data;

Node left, right;

Node(int d) {

data = d;

left = right = null;

}

}

class CodingMaxima {

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(br.readLine().trim());

while(t-->0){

String[] inputline = br.readLine().trim().split(" ");

int n = Integer.parseInt(inputline[0]);

inputline = br.readLine().trim().split(" ");

int[] arr = new int[n];

for(int i=0; i<n; i++){

arr[i] = Integer.parseInt(inputline[i]);

}

Node res = post\_order(arr, n);

printPostorder(res);

System.out.println();

}

}

// } Driver Code Ends

//User function Template for Java

//Function that constructs BST from its preorder traversal.

public static Node post\_order(int pre[], int size)

{

if(size==0)

return null;

Stack<Node> st=new Stack<>();

Node root=new Node(pre[0]);

st.push(root);

int i=1;

while(i<size ){

Node node=new Node(pre[i]);

if(node.data < st.peek().data){

st.peek().left=node;

st.push(node);

}

else{

Node parent=null;

while( !st.isEmpty() && node.data>st.peek().data ){

parent=st.pop();

}

parent.right=node;

st.push(node);

}

i++;

}

return root;

}

//{ Driver Code Starts.

public static void printInorder(Node node) {

if (node == null) {

return;

}

printInorder(node.left);

System.out.print(node.data + " ");

printInorder(node.right);

}

public static void printPostorder(Node node) {

if (node == null) {

return;

}

printPostorder(node.left);

printPostorder(node.right);

System.out.print(node.data + " ");

}

public static void printPreorder(Node node) {

if (node == null) {

return;

}

System.out.print(node.data + " ");

printPreorder(node.left);

printPreorder(node.right);

}

}

// } Driver Code Ends