**Insert a node in a BST**

**Easy**

Given a BST and a key K. If K is not present in the BST, Insert a new Node with a value equal to K into the BST.   
**Note:**If K is already present in the BST, don't modify the BST.

**Example 1:**

**Input:**

     2

  /   \

  1    3

K = 4

**Output:** 1 2 3 4

**Explanation:** After inserting the node 4

Inorder traversal will be 1 2 3 4.

**Example 2:**

**Input:**

        2

      /   \

    1     3

         \

          6

K = 4

**Output:** 1 2 3 4 6

**Explanation:** After inserting the node 4

Inorder traversal of the above tree

will be 1 2 3 4 6.

**Expected Time Complexity:** O(Height of the BST).  
**Expected Auxiliary Space:** O(Height of the BST).

**Constraints:**  
1 <= Number of nodes initially in BST <= 105  
1 <= K <= 106

**Company Tags**

[**Paytm**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Paytm) [**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung)

//{ Driver Code Starts

// Initial Template for Java

/\*package whatever //do not write package name here \*/

import java.io.\*;

import java.util.\*;

import java.math.\*;

class Node {

int data;

Node left, right;

public Node(int d) {

data = d;

left = right = null;

}

}

class CodingMaxima {

static Node buildTree(String str) {

// Corner Case

if (str.length() == 0 || str.equals("N")) return null;

String[] s = str.split(" ");

Node root = new Node(Integer.parseInt(s[0]));

Queue<Node> q = new LinkedList<Node>();

q.add(root);

// Starting from the second element

int i = 1;

while (!q.isEmpty() && i < s.length) {

// Get and remove the front of the queue

Node currNode = q.remove();

// Get the current node's value from the string

String currVal = s[i];

// If the left child is not null

if (!currVal.equals("N")) {

// Create the left child for the current node

currNode.left = new Node(Integer.parseInt(currVal));

// Push it to the queue

q.add(currNode.left);

}

// For the right child

i++;

if (i >= s.length) break;

currVal = s[i];

// If the right child is not null

if (!currVal.equals("N")) {

// Create the right child for the current node

currNode.right = new Node(Integer.parseInt(currVal));

// Push it to the queue

q.add(currNode.right);

}

i++;

}

return root;

}

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 s = br.readLine().trim();

Node root = buildTree(s);

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

Solution T = new Solution();

root = T.insert(root, key);

inorder(root);

System.out.println();

t--;

}

}

static void inorder(Node root) {

if (root == null) return;

inorder(root.left);

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

inorder(root.right);

}

}

// } Driver Code Ends

// User function Template for Java

class Solution {

// Function to insert a node in a BST.

Node insert(Node root, int key) {

if(root==null)

return new Node(key);

if(root.data>key){

root.left=insert(root.left, key);

}

else if(root.data<key){

root.right=insert(root.right,key);

}

return root;

}

}