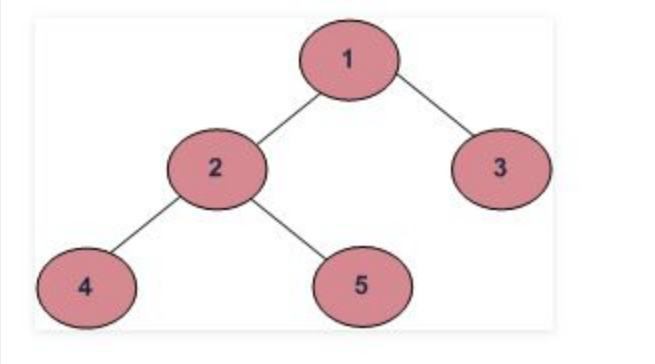
Problem Description

Given a binary tree, return the inorder traversal of its nodes' values.
For the tree given shown here,



Inorder Traversal would result in - Process (Left, Root, Right) : 4 2 5 1 3

Input format

Line1: Number of Test cases (T)
Line2 to X: First Test Case's binary tree structure (refer section below for the format)
LineX+1 to Y: Second Test Case's binary tree structure and so on.

Output format

For each test case, print on a new line, n space separated integers which are values of the inorder traversal of the nodes.

Constraints

1<= T <= 1000
1<= Number of nodes in a Tree <=10000
0 <= Value of the nodes <= 10^9
It's guaranteed that the sum of the number of tree nodes across all test cases will be less than 500000.

Sample Input 1

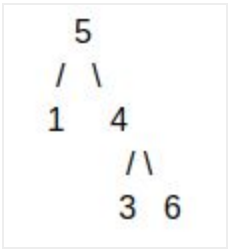
1
5
5 1 4 3 6
1 2 3
2 -1 -1
3 4 5
4 -1 -1
5 -1 -1

Sample Output 1

1 5 3 4 6

Explanation 1

Treeview



Sample Input 2

1
3
2 1 3
1 2 3
2 -1 -1
3 -1 -1

Sample Output 2

1 2 3

Explanation 2

Treeview

Instructions to create custom input for a Binary Tree

In order to specify a binary tree that can be used as custom input to the problem, you'll need to follow this convention.

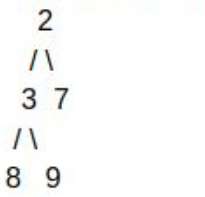
- Line 1: Number of nodes in the Binary Tree (N)
- Line 2: N space separated node values. The position of the Nodes on this line will be used to refer to them in the below lines, starting from 1.
- Line 3 to N+2: Lines specifying the child nodes for each of the N nodes

Format of each line (space separated): Parent_node Left_child_node Right_child_node

```
* Parent_node - Node at this Position on Line 2 is the Node to which
we are assigning the Left and Right child here
* Left_child_node - Node at this position on Line 2 is the left
child. Specify -1 if there is no Left child.
* Right_child_node - Node at this position on Line 2 is the right
child. Specify -1 if there is no Right child.
```

Example1

If you want to create a Tree that looks like this:



Your input would be:

5	→ Number of Nodes
2 3 7 8 9	→ Node values
1 2 3	→ Node 1(value 2) and its child nodes (left child value 3 and right child value 7)
2 4 5	→ Node 2(value 3) and its child nodes (left child value 8 and right child value 9)
3 -1 -1	→ Node 3(value 7) and its child nodes (left and right child are Null i.e. Leaf Node)
4 -1 -1	→ Node 4(value 8) and its child nodes (left and right child are Null i.e. Leaf Node)
5 -1 -1	→ Node 5(value 9) and its child nodes (left and right child are Null i.e. Leaf Node)