1 Binary tree traversals

Problem Introduction

In this problem you will implement in-order, pre-order and post-order traversals of a binary tree. These traversals were defined in the week 1 lecture on tree traversals, but it is very useful to practice implementing them to understand binary search trees better.

Problem Description

Task. You are given a rooted binary tree. Build and output its in-order, pre-order and post-order traversals.

Input Format. The first line contains the number of vertices n. The vertices of the tree are numbered from 0 to $n - 1$. Vertex 0 is the root.

The next n lines contain information about vertices 0, 1, ..., $n-1$ in order. Each of these lines contains three integers key_i, $left_i$ and $right_i$ — key_i is the key of the i-th vertex, $left_i$ is the index of the left child of the i-th vertex, and $right_i$ is the index of the right child of the i-th vertex. If i doesn't have left or right child (or both), the corresponding $left_i$ or $right_i$ (or both) will be equal to -1.

Constraints. $1 \le n \le 10^5$; $0 \le key_i \le 10^9$; $-1 \le left_i, right_i \le n - 1$. It is guaranteed that the input represents a valid binary tree. In particular, if $left_i \ne -1$ and $right_i \ne -1$, then $left_i \ne right_i$. Also, a vertex cannot be a child of two different vertices. Also, each vertex is a descendant of the root vertex.

Output Format. Print three lines. The first line should contain the keys of the vertices in the in-order traversal of the tree. The second line should contain the keys of the vertices in the pre-order traversal of the tree. The third line should contain the keys of the vertices in the post-order traversal of the tree.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time (sec)	1	1	12	6	1.5	2	6	6	12

Memory Limit. 512MB.

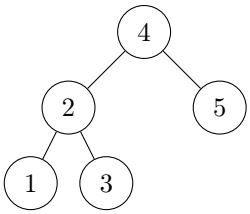
Sample 1.

Input:

```
5
4 1 2
2 3 4
5 -1 -1
1 -1 -1
3 -1 -1
```

Output:

```
1 2 3 4 5
4 2 1 3 5
1 3 2 5 4
```



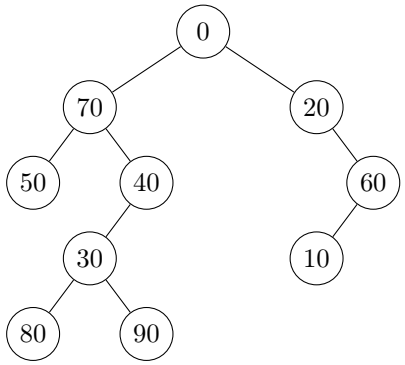
Sample 2.

Input:

```
10
0 7 2
10 -1 -1
20 -1 6
30 8 9
40 3 -1
50 -1 -1
60 1 -1
70 5 4
80 -1 -1
90 -1 -1
```

Output:

```
50 70 80 30 90 40 0 20 10 60
0 70 50 40 30 80 90 20 60 10
50 80 90 30 40 70 10 60 20 0
```



Starter Files

There are starter solutions only for C++, Java and Python3, and if you use other languages, you need to implement solution from scratch. Starter solutions read the input, define the methods to compute different traversals of the binary tree and write the output. You need to implement the traversal methods.

What to Do

Implement the traversal algorithms from the lectures. Note that the tree can be very deep in this problem, so you should be careful to avoid stack overflow problems if you're using recursion, and definitely test your solution on a tree with the maximum possible height.