Practice Arena

Practice problems aimed to improve your coding skills.

- PRACTICE-02_SCAN-PRINT
- PRACTICE-03_TYPES
- LAB-PRAC-02_SCAN-PRINT
- LAB-PRAC-01
- PRACTICE-04_COND
- BONUS-PRAC-02
- LAB-PRAC-03_TYPES
- PRACTICE-05_COND-LOOPS
- LAB-PRAC-04_COND
- LAB-PRAC-05_CONDLOOPS
- PRACTICE-07_LOOPS-ARR
- LAB-PRAC-06_LOOPS
- LAB-PRAC-07_LOOPS-ARR
- LABEXAM-PRAC-01_MIDSEM
- PRACTICE-09_PTR-MAT
- LAB-PRAC-08_ARR-STR
- PRACTICE-10_MAT-FUN
- LAB-PRAC-09_PTR-MAT
- LAB-PRAC-10_MAT-FUN
- PRACTICE-11_FUN-PTR
- LAB-PRAC-11_FUN-PTR
- LAB-PRAC-12_FUN-STRUC
- LABEXAM-PRAC-02_ENDSEM
- LAB-PRAC-13_STRUC-NUM
- LAB-PRAC-14_SORT-MISC
  - Predecessor and Successor
  - Insertion Sort
  - Link a List
  - The United Sums of Arrays
  - Bubble Sort
  - Pretty Queues Revisited
  - Just About Sorted
  - Brick Sort
  - All My Descendants
  - Mr C likes a Majority
  - Cocktail Sort
  - All My Descendants - Part II

All My Descendants - Part II

LAB-PRAC-14_SORT-MISC

All My Descendants - Part II [20 marks]

Problem Statement
We have read about binary trees in the tutorial. These are data structures where all nodes, except the root, have exactly one parent, and all nodes may have a left child and a right child. Nodes that have no children are called leaves. In this problem, we will construct a binary tree using the concepts we learnt while studying linked lists.

The first line of the input will give you n, a strictly positive integer. This will be the number of nodes in the tree. The next line will give you n integers, separated by a space. These are the occupants of a binary tree we are going to construct. However, the binary tree does not contain these elements in this order. Store these n integers in an array arr (please refer to the hint below to see how to do so).

After this, in the next line, we will give you the index of the root of the binary tree in the array arr. After this, there will be n lines telling you which are the children of these n nodes. Each line will contain a triplet of numbers a b c (i.e. the three numbers will be separated by a space). This will indicate that the element at index a in the array arr has the element at index b in the array arr, as its left child, and the element at index c in the array arr, as its right child. If a node has no left child, b will be -1. If a node has no right child, c will be -1.

Create a binary tree with n nodes with the elements in the given arrangement. Please refer to the hints to see an easy way to do so. In your output, you need to print the nodes of the tree in what is known as an *in-order traversal*. Traversal in a linked list is simple - just for from left to right!. However, in a tree, it is not clear in which order should the nodes be visited. In-order traversal is just one example of many possible ways to do so. The in-order traversal of a binary tree is recursively defined as follows.

Suppose we have a binary tree with a root which has a left child and a right child (see example below). Notice that the left child and the right child can have their own descendants. The left child and all its descendants are collectively called the left subtree of the root node. The right child and all its descendants are collectively called the right subtree of the root node. Note that the left and right subtrees are binary trees themselves.

The in-order traversal of a binary tree is defined as the in-order traversal of the left subtree of the root followed by the value stored at the root itself, followed by the in-order traversal of the right subtree of the root. The in-order traversal of a tree whose root does not have a left child is simply the value at the root followed by the in-order traversal of the right subtree of the root. The in-order traversal of a tree whose root does not have a right child is simply the in-order traversal of the left subtree of the root followed by the value at the root. The in-order traversal of a binary tree whose root has no children is simply the value at the root itself.

In your output you have to print the in-order traversal of the tree we have given you. Print each element on a different line. Be careful of the order in which you print elements. You will get partial marks only if you print the correct element at its correct location. There should be no spaces in your output in any line. Note that this definition is inherently recursive so you should use recursion to solve this problem.

P.S. The name of this problem is derived from that of a television serial called "All My Children" that used to air in the US. The serial ran for a ridiculous 41 years, spanning multiple resets and

relaunches, and finally ended in 2011

Caution

1. A node may have a left child but no right child. A node may have a right child but no left child. Leaves have neither a left child nor a right child.
2. The numbers being stored in the nodes of the tree may be positive, negative or zero. Numbers may repeat in multiple nodes of the tree. Remember, this is just a binary tree, not a binary search tree. So there is no restriction on which number may be stored in which node.
3. n may be any strictly positive integer, even 1 in which the tree will have a single node - the root.
4. We will not penalize you for extra newlines at the end of your output. However, there should be no extra spaces at the end of any of the lines in your output.

HINTS: Following these steps might make your life easier:

1. Create a structure node to store an integer and a left pointer and a right pointer.
2. Make an array of n nodes to store the n integers as given as well as the pointers to the left and the right children.
3. Store the location of the root node
4. For each triplet (a b c) given as input, make the node at index a in the array point to the node at index b in the array as its left child and the point to the node at index c in the array as its right child
5. Try to write a recursive function to print the in-order traversal of the tree.

Note that we are asking you to use a static data structure like an array, to store a binary tree, which is a dynamic data structure, only to make this problem less complicated. What you have created is not a very efficient dynamic data structure.

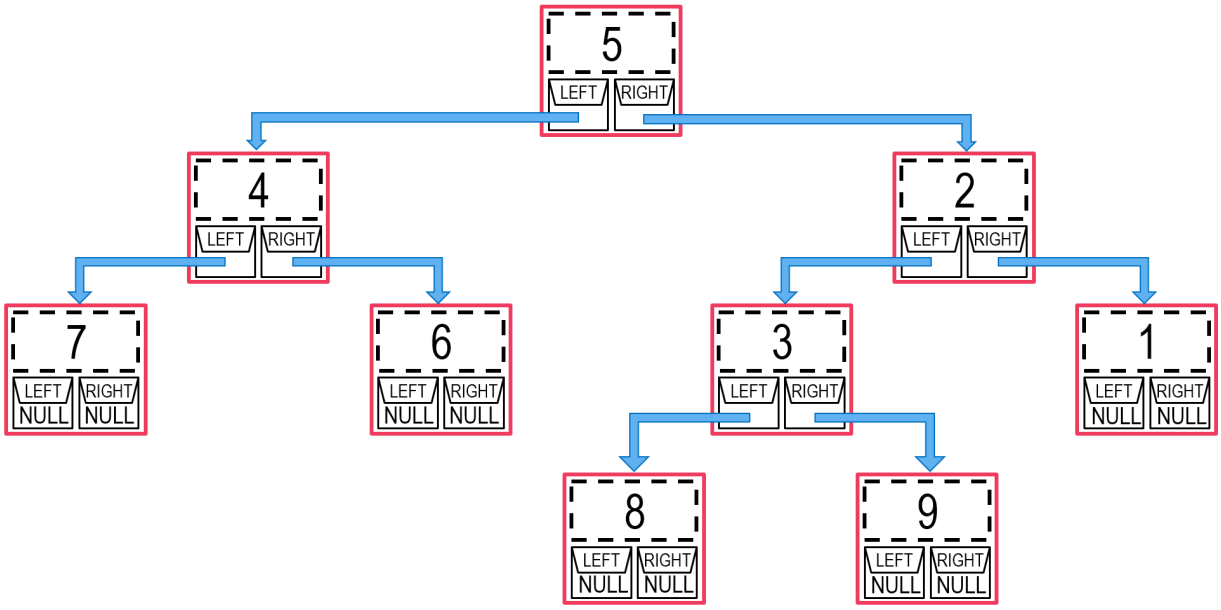
EXAMPLE:
INPUT
9
1 2 3 4 5 6 7 8 9
4
4 3 1
2 7 8
7 -1 -1
3 6 5
5 -1 -1
6 -1 -1
8 -1 -1
1 2 0
0 -1 -1

OUTPUT:
7
4
6
5

8
3
9
2
1

Explanation: See the following image to see what this tree looks like.

1. The in-order traversal of the left subtree of the root is 7 4 6 since the left subtree has 4 as the root which has its left child as 7 and right child as 6. Thus, the left subtree and the right subree of the node 4 have only node each.
2. The in-order traversal of the right subtree of the root is 8 3 9 2 1 since the right subtree has 2 as the root which has its left child as 3 and right child as 1. Thus, the left subtree of the node 2 has the in-order traversal 8 3 9 and the right subree of the node 2 has only one node.



Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. All lines carry equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6296)