Tournament Recovery

Hayden Rudson is really into sports. He likes tournament brackets (typically represented by a binary tree) that have intense match-ups. Hayden's favorite tournament brackets get mounted on a wall in a rec room in his house. Hayden's friend Leon (a recent CS graduate) likes to re arrange different data structures he sees. When Hayden was not around Leon reorganized the tournament to be a binary **SEARCH** tree. When Hayden noticed his favorite tournament was broken, Hayden had a little bit of a melt down. Feeling bad, Leon wants to now fix the tournament tree back into the original one.

Luckily, Hayden remembers a post-order traversal of the original Binary Tree (it was the play order), and Leon remembers a pre-order traversal (because he is weird). Your job is to write a program to compute the original structure. Printing the original structure can be difficult without doing image generation, so to ensure the tree structure is correct you only need to print a valid in-order traversal of the original Binary Tree.

Problem

Take in a pre- and post-order traversal compute a output a valid in-order traversal.

Input Specification

The first line of input contains a single positive integer n ($n \leq 1{,}000{,}000$). Representing the number of nodes in the tree. The next line contains a list of n space separated, distinct, positive integers between 1 and n inclusive. These integers represent the post-order of the Binary tree, that Hayden remembers. The next line contains a list of n space separated, distinct, positive integers between 1 and n inclusive. These integers represent the pre-order of the Binary tree, Leon remembers.

Output Specification

Output a single line containing a list of n space separated, distinct, positive integers between 1 and n inclusive, which represents the recovered in-order traversal of the Binary tree.

IF MULTIPLE IN-ORDER TRAVERSALS EXIST, PRINT ANY <u>ONE</u> OF THEM.

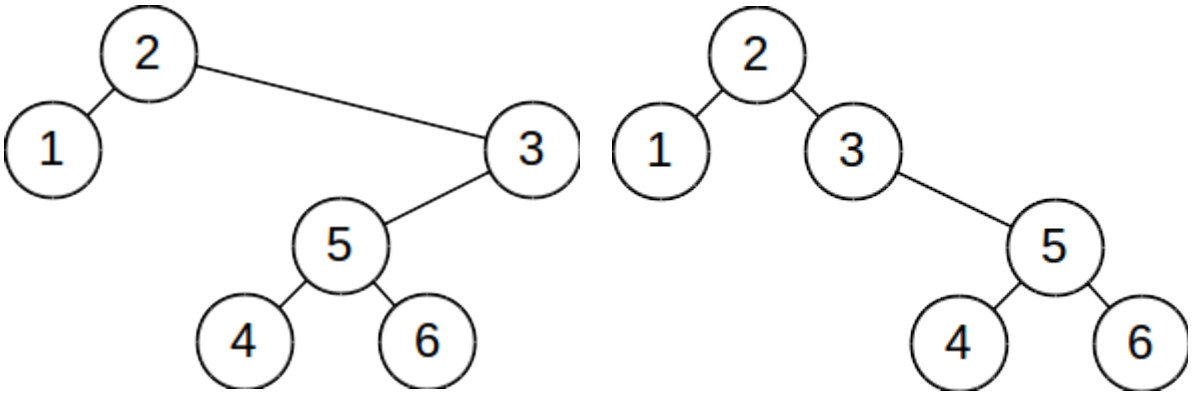
It is guaranteed that there exists at least one solution.

Sample Input	Sample Output
6 1 4 6 5 3 2 2 1 3 5 4 6	1 2 4 5 6 3
7 7 1 6 4 2 5 3 3 5 1 7 2 6 4	7 1 5 6 2 4 3
3 1 2 3 3 2 1	2 1 3

Explanation

Case 1

Here are the two possible tree structures that generates the given post and pre order traversals,



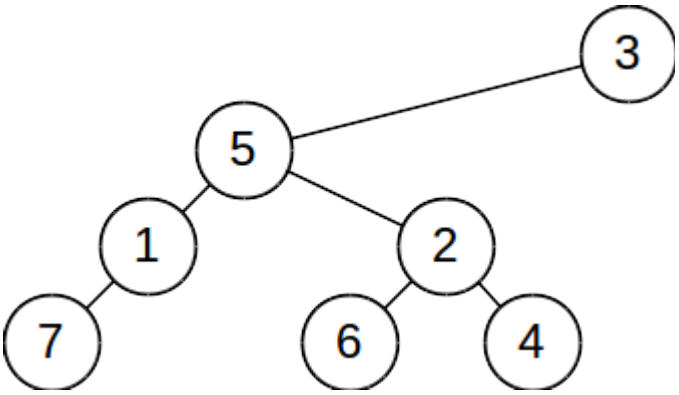
The left tree corresponds the the given output. The right tree is also acceptable, and its in order would be

1 2 3 4 5 6

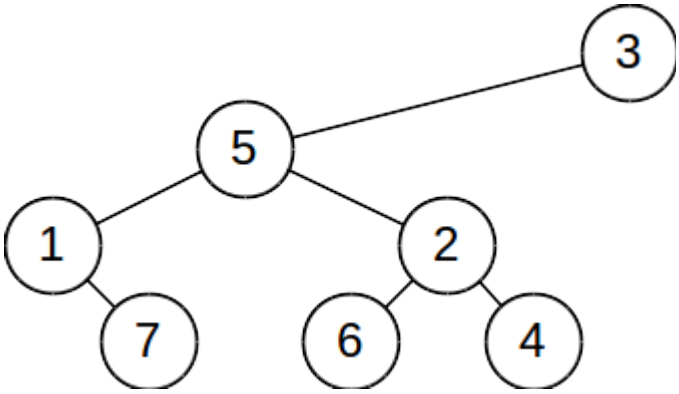
The only choice we have is whether the node 5 is a left or right child of 3. All the parent child relationships must remain as they are.

Case 2

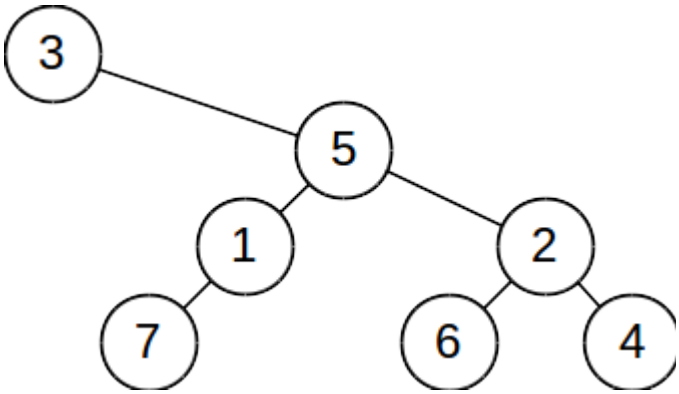
The following 4 structures would all be valid answers. The underlined one was selected



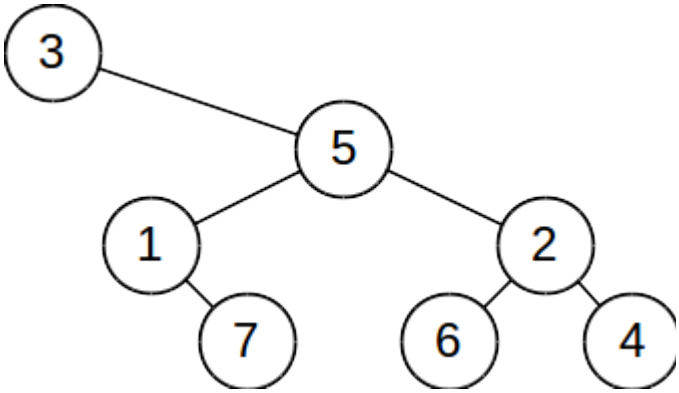
7 1 5 6 2 4 3



1 7 5 6 2 4 3



3 7 1 5 6 2 4

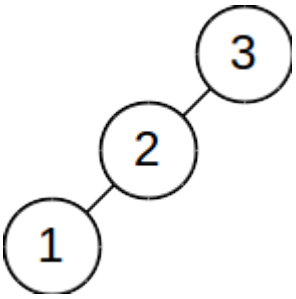


3 1 7 5 6 2 4

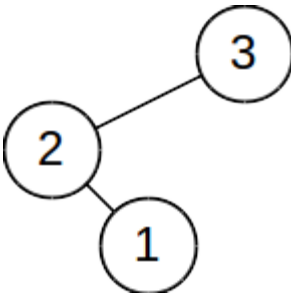
Case 3

There are 4 different answers for the 3 case.

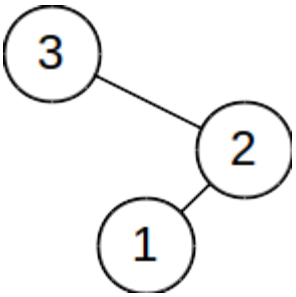
1 2 3



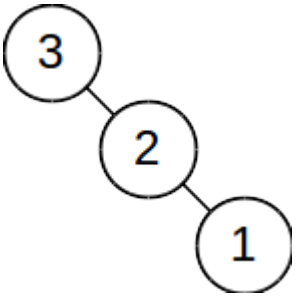
2 1 3



3 1 2



3 2 1



As long as 3 was not between 1 and 2 the order would be valid.

Solution Thoughts

I recommend handling the tree building recursively. Pass to your function indices representing which part of the pre/post traversals are under consideration for some subtree. It is a good idea to divide the traversals into two pieces (one for each child) and work with these slightly smaller sections of pre/post traversals in two different recursive calls.

Note that the child of some node (if it exists) will follow directly after the parent in the pre-order or directly before the parent in the post-order. If a node has exactly one child the two values will be the same. These values will be different if a node has two children.

Finally, if only one child exists for a node, it would be impossible to determine if the node was a left or a right child for the parent node. This results in the multiple possible answers.

Grading Details

Read/Write from/to standard input/output – 10 points

Good comments, whitespace, and variable names – 15 points

No extra input output (e.g. input prompts, “Please enter the number of words”) – 10 points

Use some way to quickly look up the position of a number within a given traversal – 10 points

Implement some method to quickly determine the children of some node – 5 points

Your program will be tested on 10 test cases – 5 points each

No points will be awarded to programs that do not compile using gcc -std=gnu11 (gnu “eleven”).

For this problem there is no limitation on what can be used.

Any case that causes your program to return a non-zero error return code will be treated as completely wrong. Additionally any case that takes longer than the maximum allowed time (the max of {5 times my solution, 5 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.

Follow the output format as specified for full points.