

Coding Assignment 2: CS2233

September 29, 2025

Kindly adhere to the following instructions.

- Please write a C/C++ program corresponding to each problem. Your code should be well commented and variable names should be appropriately chosen. Also prepare a **readme** text file where you can mention instructions to run the program/how to take input etc.
 - Create a folder and put all the code files and **readme** text file in it, give name to the folder as “your-Name_yourRollNo”, zip the folder and submit it to the google classroom portal.
 - Strictly follow the input and output format for each problem.
 - Any code that does not follow the input-output criteria won't be evaluated and will get **ZERO**.
 - Your code will also be checked against plagiarism (both from web and peer).
 - Any form of plagiarism (web/chatGPT/with peers) will be severely penalised and will result in F grade.
 - The submission (strict) timeline is 3rd October, Friday, 11 AM.
 - Each question consists of 10 marks.
-

General Instruction

The description of the *array representation of a tree* is as follows:

- Array index starts with 1.
- The left and right children of the i^{th} node present in the $2i^{th}$ and $(2i + 1)^{th}$ index, respectively.
- If let's say left child of the i^{th} node is empty then the $2i^{th}$ index of array contains *NULL*, similarly for the right child also.
- For example, See the following images.

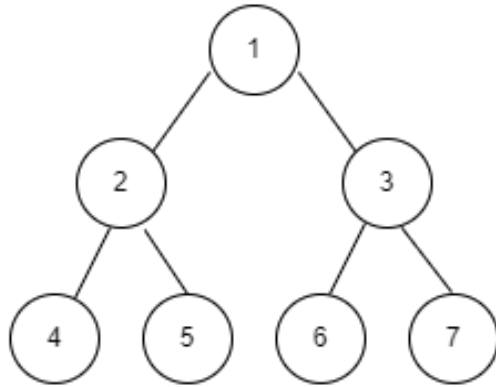
1. Write a non-recursive implementation of **inorder**, **preorder**, **postorder** traversal.

Input format

- First line will contain k , which indicates the number of test cases.
- Following k lines will contain arrays of integers, denoting the array representation of a tree.

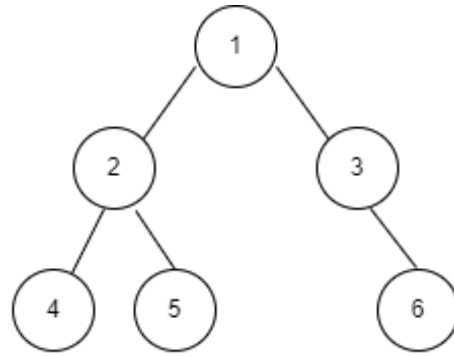
Output format

- Your output also should contain $3k$ lines, which indicates the output of corresponding k inputs.
- For each input, first line contains the **inorder** traversal, second line contains the **preorder** traversal, and third line contains the **postorder** traversal.



1	2	3	4	5	6	7
---	---	---	---	---	---	---

Array representation of the above tree



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

Array representation of the above tree

Example:

Input:

```

2
1 2 3 4 5 6 7
1 2 3 4 5 NULL 6
  
```

Output:

```

4 2 5 1 6 3 7    %inorder traversal of 1st tree%
1 2 4 5 3 6 7    %preorder traversal of 1st tree%
4 5 2 6 7 3 1    %postorder traversal of 1st tree%
4 2 5 1 3 6      %inorder traversal of 2nd tree%
1 2 4 5 3 6      %preorder traversal of 2nd tree%
4 5 2 6 3 1      %postorder traversal of 2nd tree%
  
```

Note: %comments% are there for your understanding, you need not print this.

- Write a C program that takes **inorder** and **preorder** traversal as input, output the tree. You need to print the array representation of the tree. Your code should output an error message if the **inorder** and **preorder** are not corresponding to the same tree.

Input format

- First line will contain k , which indicates the number of test cases.
- Following $2k$ lines will contain integers of each $2k$ arrays.
- $(2i - 1)^{th}$ line contains the **inorder** traversal for the i^{th} test case.
- $2i^{th}$ line contains the **preorder** traversal for the i^{th} test case.

Output format

- Your output also should contain k lines, which indicates the output of corresponding k tree inputs.
- Each line will contain the array representation of the tree as described in the general instruction.
- If the **inorder** and **preorder** are not corresponding to the same tree, then print **ERROR**.

Example:

Input:

```
3
4 2 5 1 6 3 7    %inorder traversal of 1st tree%
1 2 4 5 3 6 7    %preorder traversal of 1st tree%
4 2 5 1 3 6      %inorder traversal of 2nd tree%
1 2 4 5 3 6      %preorder traversal of 2nd tree%
4 2 6 1 3 5      %inorder traversal of 3rd tree%
1 2 4 5 3 6      %preorder traversal of 3rd tree%
```

Output:

```
1 2 3 4 5 6 7
1 2 3 4 5 NULL 6
ERROR
```

Note: %comments% are there for your understanding.

3. **Most Frequent Pair Distance in a Binary Tree** : Given a binary tree with n nodes, find the distance that occurs most frequently. If there is a tie, return the smallest distance.

Input format

- First line: integer n , the number of nodes.
- Second line: n space-separated integers for Preorder traversal.
- Third line: n space-separated integers for Inorder traversal.

Output format

- A single integer: the most frequent distance.

Expected Time Complexity : $O(n^2)$

Example:

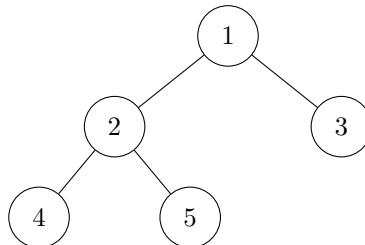
Input:

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

Output:

```
1
```

Explanation: The tree reconstructed from the traversals is shown below:



After calculating all 10 pairwise distances, the frequencies are:

- **Distance 1:** 4 times (e.g., $d(1,2)$, $d(1,3)$, $d(2,4)$, $d(2,5)$)
- **Distance 2:** 4 times (e.g., $d(1,4)$, $d(1,5)$, $d(2,3)$, $d(4,5)$)
- **Distance 3:** 2 times ($d(3,4)$, $d(3,5)$)

A tie exists between distances 1 and 2. Per the rule, the smallest tied distance is chosen.

4. **Longest Equal-Value Path in a Binary Tree :** You are given a binary tree with n nodes, where each node contains an integer value. A path is defined as a sequence of connected nodes (via edges) in the tree, where each step moves either from a parent to a child or from a child to a parent. A path is called **equal-value** if all nodes in the path have the same value.

Your task is to find the **length (number of nodes)** of the longest equal-value path in the tree.

Input Format

- The first line contains an integer n — the number of nodes.
- The next line contains n integers — the values of the nodes (1-indexed).
- The next $n - 1$ lines each contain two integers u, v representing an undirected edge between nodes u and v , where u and v are the 1-based indices of the nodes as given in the second line of input.

Output Format

Print a single integer — the length of the longest equal-value path.

Example 1

Input:

```
6
5 5 5 5 1 5
1 2
1 3
2 4
3 5
3 6
```

Output:

```
5
```

Explanation: The longest path with value 5 is $4 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 6$, containing 5 nodes.

Example 2

Input:

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

Output:

```
4
```

Explanation: The longest path with value 2 is $3 \rightarrow 1 \rightarrow 2 \rightarrow 5$, containing 4 nodes.

5. **K-th Node on the Path in a BST :** You are given a Binary Search Tree (BST) and two distinct nodes u and v . There exists a **unique path** between u and v in the BST.

Your task is to find the **k -th node on this path** (counting from u). If no such node exists, print -1 . Multiple queries (u, v, k) may be asked.

Input Format

- The first line contains an integer n — the number of nodes in the BST.
- The next line contains n integers — the node values (inserted in order into the BST).
- The next line contains an integer q — the number of queries.
- The following q lines each contain three integers: u, v, k . ($k > 0$)

Output Format

- For each query, print the k -th node on the path from u to v , or -1 if it does not exist.

Example 1

Input:

```
7
4 2 6 1 3 5 7
2
1 7 3
3 5 2
```

Output:

```
4
2
```

Example 2

Input:

```
5
10 5 15 3 7
3
3 15 4
7 5 2
10 7 4
```

Output:

```
15
5
-1
```

6. **Range Sum of BST:** You are given the root node of a Binary Search Tree (BST) and two integers low and $high$. Your task is to return the **sum of values of all nodes** with a value in the **inclusive range** $[low, high]$.

Input Format

- The first line contains the array representation of the BST.
- The next line contains two integers — **low** and **high**.

Output Format

- Print the sum of node values that lie in the range $[low, high]$.

Example 1

Input:

```
10 5 15 3 7 null 18
7 15
```

Output:

```
32
```

Explanation: Nodes 7, 10, and 15 are in the range $[7, 15]$. $\text{Sum} = 7 + 10 + 15 = 32$.

Example 2

Input:

```
10 5 15 3 7 13 18 1 null 6
6 10
```

Output:

```
23
```

Explanation: Nodes 6, 7, and 10 are in the range $[6, 10]$. $\text{Sum} = 6 + 7 + 10 = 23$.