

Lab Assignment 3: CS2233

September 6th, 2024

Question:

Given **preorder** and **inorder** traversal of a binary tree, write a program which constructs the corresponding original tree. You can assume that the input is correct **preorder** and **inorder** traversals.

You can validate the answers by reprinting their correct **preorder**, **postorder** **inorder** traversals.

Also give an asymptotic upper bound on the worst case running time.

Suggestions on how to write this code:

- Each node of the tree should use the following **struct** data type:

```
struct node
{
    int data;
    struct node *left;
    struct node *right;
};
```

- write a function **createNode (key)** that creates a node with **data** value as **key**; **left** and **right** pointers are **NULL**; and return pointer to the node
- write a function **insertLeft(root, key)** that creates a node with **data** value as **key**; and insert it as the left child of **root**. Write a similar function **insertRight(root, key)**.
- write the functions for **preorder**, **inorder** and **postorder** traversals. Each function should take only the pointer to the root node as input and perform the traversal.

Test cases:

Input format

- First line will contain **k** (the number of test cases).

- Second line will contain n for the first test case. (the number of elements in the tree).
- Third line will contain n elements of Preorder array for the first test case.
- Forth line will contain n elements of Inorder array for the first test case.
- Similarly, $3i - 1^{th}$ line will contain n for the i^{th} test case.
- $3i^{th}$ line will contain n elements of Preorder array for the i^{th} test case.
- $3i + 1^{th}$ line will contain n elements of Inorder array for the i^{th} test case.

Output format

- Your output should contain Preorder, Postorder and Inorder traversals of tree for each test cases (you have to create tree and then print these using respective tree).

Example Input

```

2
9
1 2 4 5 6 7 3 8 9
4 2 6 5 7 1 3 9 8
9
3 5 6 2 7 4 1 0 8
6 5 7 2 4 3 0 1 8

```

Output

```

1 2 4 5 6 7 3 8 9
4 6 7 5 2 9 8 3 1
4 2 6 5 7 1 3 9 8

3 5 6 2 7 4 1 0 8
6 7 4 2 5 0 8 1 3
6 5 7 2 4 3 0 1 8

```