Indian Institute of Technology Ropar

Data Structures and Algorithms

CS506 and CS205

Lab Assignment 3

[Due Date: 23rd September'2024 **]**

Problem Statement:

You must implement programs for Binary Tree, Binary Search Tree (BST), and AVL Tree. There will be certain operations you need to implement for the given trees. The operations will be selected using some key inputs, mentioned below.

Key (Query Identifier)	Input	Functionality
B	Sequence of integers N A1 A2 A3 ... AN	Constructing BST N: N operations including insertion/deletion Ai: Operand If Ai is positive => Insert in BST Else if Ai is negative => Remove it from BST Else if Ai is 0 => No operation (For deletion, suppose the input is -5, then you need to delete 5 from the tree (if it exists).
A	Sequence of integers N A1 A2 A3 ... AN	Constructing AVL N: N operations including insertion/deletion Ai: Operand If Ai is positive => Insert in AVL Else if Ai is negative => Remove it from AVL Else if Ai is 0 => No operation (For deletion, suppose the input is -5, then you need to delete 5 from the tree (if it exists).

I	A1 (A1 will be positive value)	Insert a node A1 in the current existing tree. //If the tree is empty, add the element as the first node of a BST
R	A1 (A1 will be positive value)	Remove the node A1 from the current existing tree //If the element is not there or the tree is empty, do nothing
F	X	Find X in the current tree Print “Yes” if X is in Tree Print “No” if X is not in the Tree //If X is negative or 0, then do nothing
L		Print the number of Leaves present in the tree
N		Print the number of total nodes in the tree
Q		Print In-order traversal
W		Print Pre-order traversal
E		Print Post-order traversal
H		Print the Height of the tree (Single node will have height 0)
M		Print the Boundary Traversal (Clockwise) (Definition and reference have been mentioned below)

C	A1 A2	Print the lowest common ancestor of node A1 and node A2.
		
		//If node A1 or node A2 does not exist, print -1.
		//if A1==A2 print A1.

Z	N
	A1 A2 A3 A4 ...AN
	B1 B2 B3 B4 ... BN
		//N: Number of elements
		//A1 ... AN : Preorder sequence of a tree
		//B1 ... BN : Inorder Sequence of a tree
		
		Print the Post order sequence of a tree
		C1 C2 C3 C4 ... CN (only space separated)

Y	N
	A1 A2 A3 A4 ...AN
	B1 B2 B3 B4 ... BN
		//N: Number of elements
		//A1 ... AN : Postorder sequence of a tree
		//B1 ... BN : Inorder Sequence of a tree
		
		Print the Pre order sequence of a tree
		C1 C2 C3 C4 ... CN (only space separated)

K		Remove the whole tree if it exists.
D		End of operations in the current Test Case and go to next Test Case (if any)

General Assumptions/Constraints:

1. All the nodes of a tree must have a positive number.
2. If the input is 0, perform nothing.
3. For B and A, negative numbers are for deleting the node having the same value with a positive sign of the input.
4. **Do not print any prompt** except “Yes”/”No” for searching.
5. **By default, BST will be constructed.** It will be helpful if the input is “I” while no tree exists at that moment.
6. There will be no output for B, A, I, R, and K.
7. For Z and Y, print the traversal using a single white space.
8. For Z and Y, the given Pre/Post-order and Inorder will be independent Binary Trees i.e. it is not related to the current existing binary/AVL tree.
9. Each output will be in a single line. Once it is done, go to the immediate next line. No extra space, tab, lines.
10. **While inserting elements in the tree, if there is a duplicate element, perform nothing. That means the tree will not have duplicate elements present.**
11. **While deleting, if there are two child nodes of the key node, you will follow the policy of In-order successor replacement.**

Input Format:

The first line will be the number of Test Cases.

Then for each test case, operations will be performed until D comes. Each test case will end only when D comes.

Sample Input:

```
3
B
5 1 2 3 -2 4
F
3
F
2
H
K
B
4 5 -3 2 10
W
D
B
6 10 3 4 0 -4 15
R
0
Q
N
L
C
3 15
M
D
Z
6
5 2 1 3 8 6
1 2 3 5 6 8
I
10
E
D
```

Expected Output:

```
Yes
No
2
5 2 10
3 10 15
3
2
10
10 15 3
1 3 2 6 8 5
10
```

Boundary Traversal Reference:

Given a binary tree, print boundary nodes of the binary tree Clockwise starting from the root.

https://www.geeksforgeeks.org/boundary-traversal-of-binary-tree/

Programming Languages Allowed:

C/C++

(Using any kind of inbuilt libraries/containers for the data structures is strictly forbidden. If found, then penalties will be imposed)

Expected File(s):

One single file of .c / .cpp

Naming Convention of File

FirstName_EntryNo_CS506_Lab03_Prog.c (or .cpp)

**Students of CS205 will write CS205 instead of CS506*

Example:-

Rejoy_2023CSM1011_CS506_Lab03_Prog.c (or .cpp)

Instructions for Programming:

- Don’t put unnecessary pieces of information or codes.
- Good documentation in the codes is highly appreciated. (You may use comments)
- Please remember to make modular codes.

Plagiarism Checking

Do not copy from each other. Plagiarism will be checked. After a certain threshold, a high penalty or even an F grade can be imposed, depending on the plagiarism rate.

Feel Free To Contact The TAs:

`Rejoy Chakraborty`
`Madhav Mishra`
`Shradha Sharma`