CSE 4304-Data Structures Lab. Winter 2022-23
Date: 17 October 2023
Target Group: SWE B
Topic: Binary Tree
Instructions:

- Regardless of when you finish the tasks in the lab, you have to submit the solutions in the Google Classroom. The deadline will always be at 11.59 PM of the day in which the lab has taken place.
- Task naming format: <fullID>_<Task><Lab><Group>.c/cpp. Example: 170041034_T01L02A.cpp
- If you find any issues in the problem description/test cases, comment in the google classroom.
- If you find any test case that is tricky that I didn't include but others might forget to handle, please comment! I'll be happy to add.
- Use appropriate comments in your code. This will help you to easily recall the solution in the future.
- Obtained marks will vary based on the efficiency of the solution.
- Do not use the <bits/stdc++.h> library.

Structure of a node of a binary Tree:

```
struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode() : val(0), left(nullptr), right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
}
```

Task:1

Implement the basic operations of Binary tree. Your program should include the following functions:

1. **Insert**:
   - Assuming each node contains a unique value.
   - Input starts with a number N (representing the number of nodes), followed by N lines containing the info of node.
   - Every line contains three values 'data', 'parent' and '1(left child) or 2(right child)'. For the root, the second and third parameter is Null (0).
   - After successful insertion, print the entire tree using preorder traversal. Beside each node, print the info of it's parent as well.
2. **Removal**: if an internal node is removed, its subtrees are also removed.
3. **Search**: Returns the node if it is present and prints it's description.
4. **Height**: Returns the height of the tree, where height of a leaf node is 0.

Input	**Output**
7 1 0 0 2 1 1 3 1 2 5 2 2 4 2 1 6 3 1 7 3 2	1(N) 2(1) 4(2) 5(2) 3(1) 6(3) 7(3) (preorder) **Note**: The tree looks like as follows: 1 / \ 2 3 / \ / \ 4 5 6 7
3 3 (search 3)	Present, Left(6), Right(7)
2 6 (remove 6)	1(N) 2(1) 4(2) 5(2) 3(1) 7(3) (preorder)
3 1 (search 1)	Present, Left(2), Right(3)
3 5	Present, Left(null), Right(null)
3 6	Not present
3 3	Present, Left(null), Right(7)
4	2
2 3	1(N) 2(1) 4(2) 5(2) (preorder)
4	2
3 3	Not present
2 2	1(N) (preorder)

4	0

Task:2

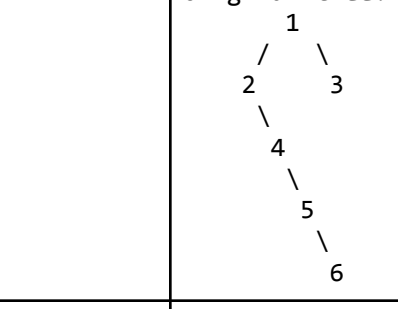
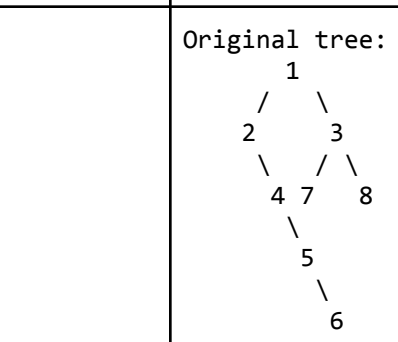
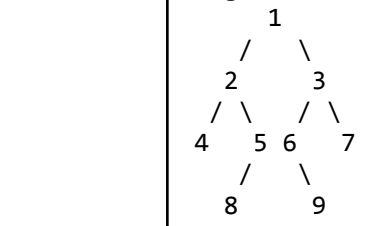
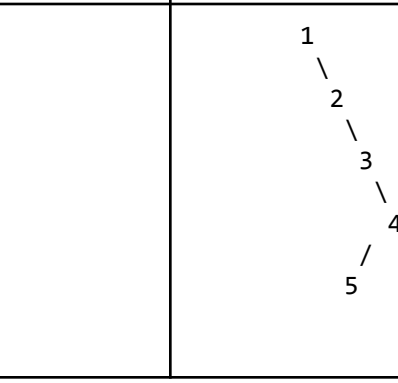
For a binary tree, show the result of level-order, inorder, preorder and postorder traversal. Input policy is the same as Task-1.

Input	Output	Commnet
5 A B A 1 C A 2 D B 1 E B 2	A B C D E (level order) D B E A C (Inorder) A B D E C (Preorder) D E B C A (Postorder)	Original Tree A / \ B C / \ D E
9 1 0 0 2 1 1 3 1 2 5 2 2 6 3 1 7 3 2 4 2 1 9 6 2 8 5 1	1 2 3 4 5 6 7 8 9 (level order) 4 2 8 5 1 6 9 3 7 (Inorder) 1 2 4 5 8 3 6 9 7 (Preorder) 4 8 5 2 9 6 7 3 1 (Postoder)	Original tree: 1 / \ 2 3 / \ / \ 4 5 6 7 / \ 8 9

Task:3

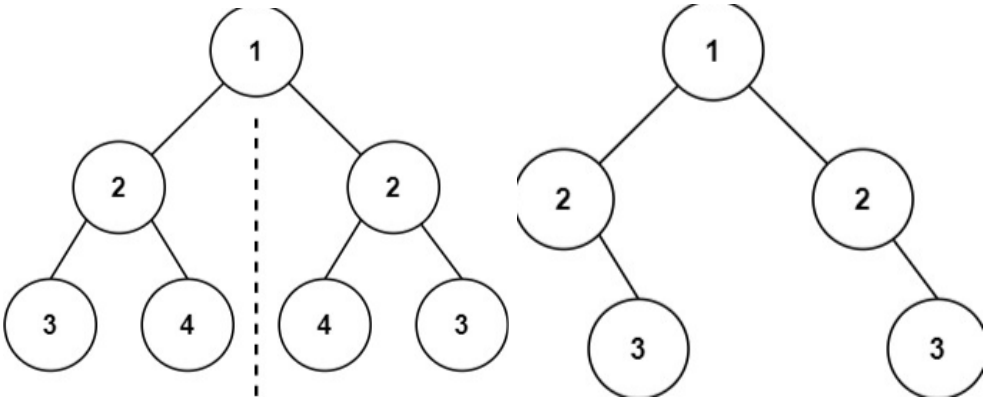
Given a binary tree with N nodes, your task is to print all the paths from the root to the leaves along with the path length. You can print the paths in any order. Make sure the path starts from the root and ends at a leaf. Follow the input policy as Task 1.

Input	Output	Comment
7 1 0 0 3 1 2 2 1 1 4 2 1 5 2 2 7 3 2	1 2 4 (3) 1 2 5 (3) 1 3 6 (3) 1 3 7 (3)	Original tree: 1 / \ 2 3 / \ / \ 4 5 6 7

6 3 1		
6 1 0 0 2 1 1 3 1 2 4 2 2 5 4 2 6 5 2	1 2 4 5 6 (5) 1 3 (2)	Original tree: 
8 1 0 0 2 1 1 3 1 2 8 3 2 4 2 2 7 3 1 5 4 2 6 5 2	1 2 4 5 6 (5) 1 3 7 (3) 1 3 8 (3)	Original tree: 
9 1 0 0 2 1 1 3 1 2 5 2 2 6 3 1 7 3 2 4 2 1 9 6 2 8 5 1	1 2 4 (3) 1 2 5 8 (4) 1 3 6 9 (4) 1 3 7 (3)	Original tree: 
7 1 0 0 2 1 2 3 2 2 4 3 2 5 4 1 6 4 2 7 6 1	1 2 3 4 5 (5) 1 2 3 4 6 7 (6)	

Task:4

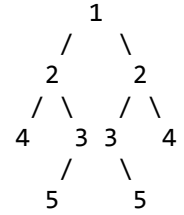
Given a binary tree with N nodes, your task is to check if it is symmetric around the root node. That is, the left subtree of the root node is the mirrored version of the right subtree of the root node.



Symmetric Tree Non Symmetric Tree

Input: First line will contain an integer N denoting the number nodes in the binary tree. Next line will contain N space separated by integers denoting the level order traversal. A node containing value -1 represents a null node.

Input	**Output**	**Comment**
7 1 2 2 4 3 3 4	Yes	Original tree: 1 / \ 2 2 / \ / \ 4 3 3 4
7 1 2 2 -1 3 -1 3	No	Original tree: 1 / \ 2 2 \ \ 3 3
15 1 2 3 -1 4 -1 -1 -1 -1 -1 5 -1 -1 -1 -1	No	Original tree: 1 / \ 2 3 \ 4 \ 5

15 1 2 2 4 3 3 4 -1 -1 5 -1 -1 5 -1 -1	Yes	Original tree: 
15 1 2 2 4 3 3 4 5 -1 -1 -1 -1 -1 -1 5	Yes	Original tree: