Generic Trees

1. **Sum of all nodes**

**Send Feedback**

Given a generic tree, count and return the sum of all nodes present in the given tree.

**Input format :**

Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output Format :**

Sum of all nodes

**Sample Input :**

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output :**

190

1. **Node With Largest Data**

**Send Feedback**

Given a generic tree, find and return the node with maximum data. You need to return the complete node which is having maximum data.

Return null if tree is empty.

**Input format :**

Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output Format :**

Node with largest data

**Sample Input :**

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output :**

50

1. **Height Of Tree**

**Send Feedback**

Given a generic tree, find and return the height of given tree.

**Input format :**

Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output Format :**

Height

**Sample Input :**

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output :**

3

1. **Print Levelwise**

**Send Feedback**

Given a generic tree, print the input tree in level wise order. ####For printing a node with data N, you need to follow the exact format -

N:x1,x2,x3,...,xn

wherer, N is data of any node present in the binary tree. x1, x2, x3, ...., xn are the children of node N

There is no space in between.

You need to print all nodes in the level order form in different lines.

**Input format :**

Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output Format :**

Level wise print

**Sample Input :**

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output :**

10:20,30,40

20:40,50

30:

40:

40:

50:

Assignment

1. **Contains x**

**Send Feedback**

Given a generic tree and an integer x, check if x is present in the given tree or not. Return true if x is present, return false otherwise.

**Input format :**

Line 1 : Integer x

Line 2 : Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output format :**

true or false

**Sample Input 1 :**

40

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 1 :**

true

**Sample Input 2 :**

4

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 2:**

false

1. **Count leaf Nodes**

**Send Feedback**

Given a generic tree, count and return the number of leaf nodes present in the given tree.

**Input format :**

Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output Format :**

Count of leaf nodes

**Sample Input 1 :**

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 1 :**

4

1. **Node with maximum child sum**

**Send Feedback**

Given a tree, find and return the node for which sum of data of all children and the node itself is maximum. In the sum, data of node itself and data of immediate children is to be taken.

**Input format :**

Line 1 : Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output format :**

Node with maximum sum.

**Sample Input 1 :**

5 3 1 2 3 1 15 2 4 5 1 6 0 0 0 0

**Sample Output 1 :**

1

1. **Structurally identical**

**Send Feedback**

Given two Generic trees, return true if they are structurally identical i.e. they are made of nodes with the same values arranged in the same way.

**Input format :**

Line 1 : Tree 1 elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

Line 2 : Tree 2 elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output format :**

true or false

**Sample Input 1 :**

10 3 20 30 40 2 40 50 0 0 0 0

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 1 :**

true

**Sample Input 2 :**

10 3 20 30 40 2 40 50 0 0 0 0

10 3 2 30 40 2 40 50 0 0 0 0

**Sample Output 2:**

false

1. **Next larger**

**Send Feedback**

Given a generic tree and an integer n. Find and return the node with next larger element in the Tree i.e. find a node with value just greater than n.

**Return NULL if no node is present with the value greater than n.**

**Input Format :**

Line 1 : Integer n

Line 2 : Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Output Format :**

Node with value just greater than n.

**Sample Input 1 :**

18

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 1 :**

20

**Sample Input 2 :**

21

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 2:**

30

1. **Replace with depth**

**Send Feedback**

In a given Generic Tree, replace each node with its depth value. You need to just update the data of each node, no need to return or print anything.

**Input format :**

Line 1 : Elements in level order form separated by space (as per done in class). Order is -

Root\_data, n (No\_Of\_Child\_Of\_Root), n children, and so on for every element

**Sample Input 1 :**

10 3 20 30 40 2 40 50 0 0 0 0

**Sample Output 1 : (Level wise, each level in new line)**

0

1 1 1

2 2