Binary Trees – 1

1. **Binary Tree Node**

**Send Feedback**

A node in Binary trees can have how many children ?

1. 2
2. Any number of children
3. 0 or 1 or 2 answer
4. 0 or 1
5. **Input Binary Tree**

**Send Feedback**

Consider the below code

def takeInput():

print("Enter root data")

rootData = int(input())

if (rootData == -1) {

return None

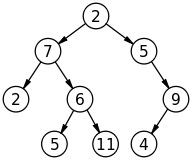
}

root = BinaryTreeNode(rootData)

root.left = takeInput()

root.right = takeInput()

return root



What will be the input(excluding -1) to above code to construct this tree ?

1) 2 7 2 6 5 11 5 9 4

2) 2 7 5 6 11 2 5 4 9

3) 2 7 5 2 6 9 5 11 4

1. 1 answer
2. 2
3. 3
4. **Sum Of Nodes**

**Send Feedback**

#### For a given Binary Tree of integers, find and return the sum of all the nodes data.

##### Example:

10

/ \

20 30

/ \

40 50

When we sum up all the nodes data together, [10, 20, 30, 40 50] we get 150. Hence, the output will be 150.

##### Input Format:

The first and the only line of input will contain the nodes data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The first and the only line of output prints the sum of all the nodes data present in the binary tree.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Constraints:

1 <= N <= 10^6

Where N is the total number of nodes in the binary tree.

Time Limit: 1 sec

##### Sample Input 1:

2 3 4 6 -1 -1 -1 -1 -1

##### Sample Output 1:

15

##### Sample Input 2:

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

##### Sample Output 2:

28

1. **Preorder Binary Tree**

**Send Feedback**

#### For a given Binary Tree of integers, print the pre-order traversal.

##### Input Format:

The first and the only line of input will contain the nodes data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The only line of output prints the pre-order traversal of the given binary tree.

##### Constraints:

1 <= N <= 10^6

Where N is the total number of nodes in the binary tree.

Time Limit: 1 sec

##### Sample Input 1:

5 6 10 2 3 -1 -1 -1 -1 -1 9 -1 -1

##### Sample Ouptut 1:

5 6 2 3 9 10

##### Sample Input 2:

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

##### Sample Ouptut 2:

1 2 4 5 3 6 7

1. **Postorder Binary Tree**

**Send Feedback**

#### For a given Binary Tree of integers, print the post-order traversal.

##### Input Format:

The first and the only line of input will contain the node data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The only line of output prints the post-order traversal of the given binary tree.

##### Constraints:

1 <= N <= 10^6

Where N is the total number of nodes in the binary tree.

Time Limit: 1 sec

##### Sample Input 1:

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

##### Sample Output 1:

4 5 2 6 7 3 1

##### Sample Input 2:

5 6 10 2 3 -1 -1 -1 -1 -1 9 -1 -1

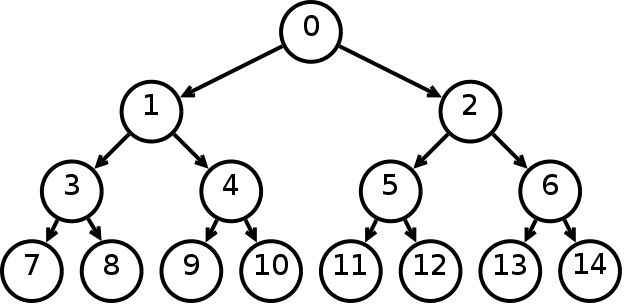
##### Sample Output 1:

2 9 3 6 10 5

1. **Binary Tree Levelwise**

**Send Feedback**

Print given tree level wise



1) 0 1 3 7 8 4 9 10 2 5 11 12 6 13 14

2) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

3) 0 2 1 6 5 4 3 14 13 12 11 10 9 8 7

1. 1
2. 2 answer
3. 3
4. **Nodes Greater Than X**

**Send Feedback**

#### For a given a binary tree of integers and an integer X, find and return the total number of nodes of the given binary tree which are having data greater than X.

##### Input Format:

The first line of input will contain the node data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

The second line of input contains an integer, denoting the value of X.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Output Format:

The only line of output prints the total number of nodes where the node data is greater than X.

##### Constraints:

1 <= N <= 10^5

Where N is the total number of nodes in the binary tree.

Time Limit: 1 sec

##### Sample Input 1:

1 4 2 3 -1 -1 -1

2

##### Sample Output 1:

2

##### Explanation for Sample Input 1:

Out of the four nodes of the given binary tree, [3, 4] are the node data that are greater than X = 2.

##### Sample Input 2:

5 6 10 2 3 -1 -1 -1 -1 -1 9 -1 -1

5

##### Sample Output 2:

3

1. **Height Of Tree**

**Send Feedback**

###### The maximum and minimum number of nodes in a binary tree of height 6 are

A tree with zero node has height 0 A tree with one node has height 1

1. 63 and 6, respectively answer
2. 64 and 5, respectively
3. 32 and 6, respectively
4. 31 and 5, respectively
5. **Height Of Tree**

**Send Feedback**

#### For a given Binary Tree of integers, find and return the height of the tree.

##### Example:

10

/ \

20 30

/ \

40 50

Height of the given tree is 3.

Height is defined as the total number of nodes along the longest path from the root to any of the leaf node.

##### Input Format:

The first and the only line of input will contain the node data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The first and the only line of output prints the height of the given binary tree.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Constraints:

0 <= N <= 10^5

Where N is the total number of nodes in the binary tree.

Time Limit: 1 sec

##### Sample Input 1:

10 20 30 40 50 -1 -1 -1 -1 -1 -1

##### Sample Output 1:

3

##### Sample Input 2:

3 -1 -1

##### Sample Output 2:

1

1. **Replace Node With Depth**

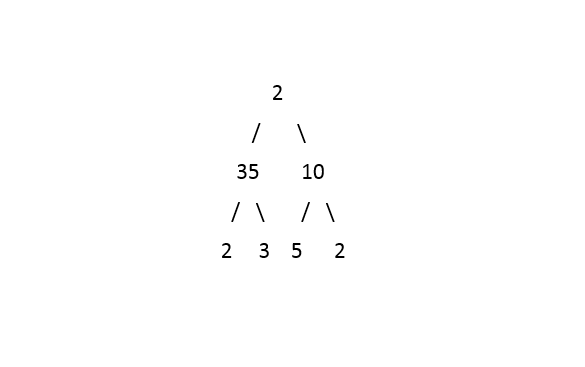
**Send Feedback**

#### For a given a Binary Tree of integers, replace each of its data with the depth of the tree.

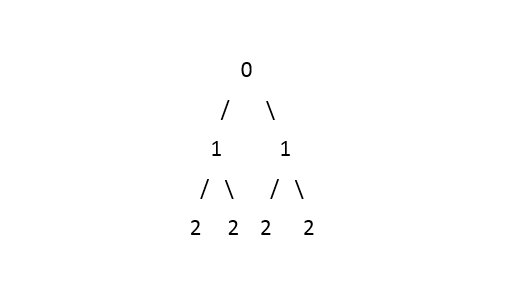
#### Root is at depth 0, hence the root data is updated with 0. Replicate the same further going down the in the depth of the given tree.

#### The modified tree will be printed in the in-order fashion.

##### Example:



The above tree after updating will look like this:



Output: 2 1 2 0 2 1 2 (printed in the in-order fashion)

##### Input Format:

The first and the only line of input will contain the node data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The first and the only line of output prints the updated tree in the in-order fashion.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Constraints:

1 <= N <= 10^5

Where N is the total number of nodes in the binary tree.

Time Limit: 1sec

##### Sample Input 1:

2 35 10 2 3 5 2 -1 -1 -1 -1 -1 -1 -1 -1

##### Sample Output 1:

2 1 2 0 2 1 2

##### Sample Input 2:

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

##### Sample Output 2:

2 1 2 0 2 1 2

Assignment

1. **Is node present?**

**Send Feedback**

#### For a given Binary Tree of type integer and a number X, find whether a node exists in the tree with data X or not.

##### Input Format:

The first and the only line of input will contain the node data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The only line of output prints 'true' or 'false'.

##### Note:

You are not required to print anything explicitly. It has already been taken care of.

##### Constraints:

1 <= N <= 10^5

Where N is the total number of nodes in the binary tree.

Time Limit: 1 sec

##### Sample Input 1:

8 3 10 1 6 -1 14 -1 -1 4 7 13 -1 -1 -1 -1 -1 -1 -1

7

##### Sample Output 1:

true

##### Sample Input 2:

2 3 4 -1 -1 -1 -1

10

##### Sample Output 2:

false

1. **Nodes without sibling**

**Send Feedback**

#### For a given Binary Tree of type integer, print all the nodes without any siblings.

##### Input Format:

The first and the only line of input will contain the node data, all separated by a single space. Since -1 is used as an indication whether the left or right node data exist for root, it will not be a part of the node data.

##### Output Format:

The only line of output prints the node data in a top to down fashion with reference to the root.

Node data in the left subtree will be printed first and then the right subtree.

A single space will separate them all.

##### Constraints:

1 <= N <= 10^5

Where N is the total number of nodes in the binary tree.

Time Limit: 1 second

##### Sample Input 1:

5 6 10 2 3 -1 -1 -1 -1 -1 9 -1 -1

##### Sample Output 1:

9

##### Sample Input 2:

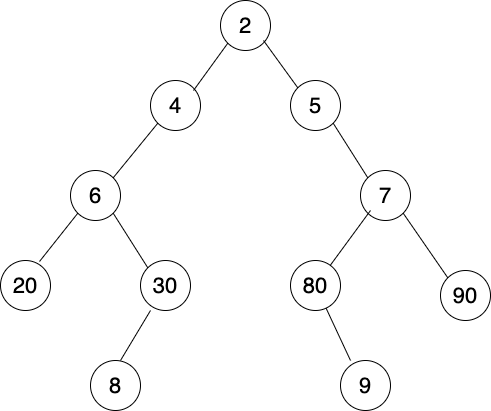
2 4 5 6 -1 -1 7 20 30 80 90 -1 8 9 -1 -1 -1 -1 -1 -1 -1 -1 -1

##### Sample Output 2:

6 8 7 9

##### Explanation of Sample Input 2:

The input tree when represented in a two-dimensional plane, it would look like this:



In respect to the root, node data in the left subtree that satisfy the condition of not having a sibling would be 6 and 8, taken in a top-down sequence. Similarly, for the right subtree, 7 and 9 are the node data without any sibling.

Since we print the siblings in the left-subtree first and then the siblings from the right subtree, taken in a top-down fashion,