**Q) Print Level Wise**

#### Given a binary tree, print the tree in level wise order.

#### For printing a node with data N, you need to follow the exact format -

N:L:x,R:y

#### wherer, N is data of any node present in the binary tree. x and y are the values of left and right child of node N. Print -1. if any child is null.

#### 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)

(If any node does not have left or right child, take -1 in its place)

##### Sample Input :

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

##### Sample Output :

8:L:3,R:10

3:L:1,R:6

10:L:-1,R:14

1:L:-1,R:-1

6:L:4,R:7

14:L:13,R:-1

4:L:-1,R:-1

7:L:-1,R:-1

13:L:-1,R:-1

Q) **Is node present ?**

#### Given a Binary Tree and an integer x, check if node with data x is present in the input binary tree or not. Return true or false.

##### Input format :

Line 1 : Elements in level order form (separated by space)

(If any node does not have left or right child, take -1 in its place)

Line 2 : Integer x

##### Output Format :

true or false

##### Sample Input :

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

7

##### Sample Output :

true

Q) **Height of binary tree**

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

##### Input format :

Nodes in the level order form (separated by space). If any node does not have left or right child, take -1 in its place

##### Output format :

Height

##### Constraints :

#### 1 <= N <= 10^5

##### Sample Input :

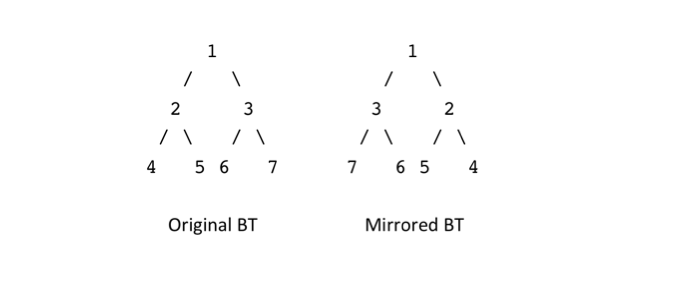
10 9 4 -1 -1 5 8 -1 6 -1 -1 3 -1 -1 -1

##### Sample Output :

5

Q) **Mirror a Binary tree**

#### Mirror the given binary tree. That is, right child of every nodes should become left and left should become right.



##### Note : You don't need to print or return the tree, just mirror it.

Input format :

Line 1 : Elements in level order form (separated by space)

(If any node does not have left or right child, take -1 in its place)

Output format : Elements in level order form (Every level in new line)

##### Sample Input 1:

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

##### Sample Output 1:

1

3 2

7 6 5 4

Q) **Preorder Binary Tree**

#### Given a binary tree, print the preorder traversal of given tree.

#### Pre-order traversal is: Root LeftChild RightChild

##### Input format :

Elements in level order form (separated by space)

(If any node does not have left or right child, take -1 in its place)

##### Output Format :

Pre-order traversal, elements separated by space

##### Sample Input :

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

##### Sample Output :

8 3 1 6 4 7 10 14 13

Q) **Postorder Binary Tree**

#### Given a binary tree, print the postorder traversal of given tree.

#### Post-order traversal is: LeftChild RightChild Root

##### Input format :

Elements in level order form (separated by space)

(If any node does not have left or right child, take -1 in its place)

##### Output Format :

Post-order traversal, elements separated by space

##### Sample Input :

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

##### Sample Output :

1 4 7 6 3 13 14 10 8

Q) **Construct Tree from preorder & inorder**

#### Given Preorder and Inorder traversal of a binary tree, create the binary tree associated with the traversals.You just need to construct the tree and return the root.

##### Note: Assume binary tree contains only unique elements.

Input format :

Line 1 : n (Total number of nodes in binary tree)

Line 2 : Pre order traversal

Line 3 : Inorder Traversal

Output Format :

Elements are printed level wise, each level in new line (separated by space).

##### Sample Input :

12

1 2 3 4 15 5 6 7 8 10 9 12

4 15 3 2 5 1 6 10 8 7 9 12

##### Sample Output :

1

2 6

3 5 7

4 8 9

15 10 12

Q) **Construct Tree from postorder & inorder**

#### Given Postorder and Inorder traversal of a binary tree, create the binary tree associated with the traversals.You just need to construct the tree and return the root.

##### Note: Assume binary tree contains only unique elements.

Input format :

Line 1 : n (Total number of nodes in binary tree)

Line 2 : Post order traversal

Line 3 : Inorder Traversal

Output Format :

Elements are printed level wise, each level in new line (separated by space).

##### Sample Input :

8

8 4 5 2 6 7 3 1

4 8 2 5 1 6 3 7

##### Sample Output :

1

2 3

4 5 6 7

8

Q) **Sum of all nodes**

#### Given a binary tree, find and return the sum of all nodes.

Input format :

Elements in level order form (separated by space). If any node does not have left or right child, take -1 in its place.

##### Sample Input :

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

##### Sample Output :

35

Q) **is Balanced**

#### Given a binary tree, check if its balanced i.e. depth of left and right subtrees of every node differ by at max 1. Return true if given binary tree is balanced, false otherwise.

Input format :

Elements in level order form (separated by space). If any node does not have left or right child, take -1 in its place.

##### Sample Input 1 :

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

##### Sample Output 1 :

false

##### Sample Input 2 :

1 2 3 -1 -1 -1 -1

##### Sample Output 2 :

true

Q) **Level order traversal**

#### Given a binary tree, print the level order traversal. Make sure each level start in new line.

Input format :

Elements in level order form (separated by space). If any node does not have left or right child, take -1 in its place.

Output Format :

Elements are printed level wise, each level in new line (separated by space).

##### Sample Input :

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

##### Sample Output :

5

6 10

2 3

9

**Q) Remove Leaf nodes**

#### Remove all leaf nodes from a given Binary Tree. Leaf nodes are those nodes, which don't have any children.

Input format :

Elements in level order form (separated by space)

(If any node does not have left or right child, take -1 in its place)

Output Format :

Elements are printed level wise, each level in new line (separated by space).

##### Sample Input :

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

##### Sample Output :

8

3 10

6 14

Q) **Level wise linkedlist**

#### Given a binary tree, write code to create a separate linked list for each level. You need to return the array which contains head of each level linked list.

Input format :

Elements in level order form (separated by space). If any node does not have left or right child, take -1 in its place.

Output format : Each level linked list is printed in new line (elements separated by space).

##### Sample Input :

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

##### Sample Output :

5

6 10

2 3

9

Q) **ZigZag tree**

#### Given a binary tree, print the zig zag order i.e print level 1 from left to right, level 2 from right to left and so on. This means odd levels should get printed from left to right and even level right to left.

Input format :

Elements in level order form (separated by space)

(If any node does not have left or right child, take -1 in its place)

Output Format :

Elements are printed level wise, each level in new line (separated by space).

##### Sample Input :

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

##### Sample Output :

5

10 6

2 3

9

Q) **Nodes without sibling**

#### Given a binary tree, print all nodes that don’t have a sibling.

##### Edit : Print the elements in different lines. And order of elements doesn't matter.

##### Input format :

Elements in level order form (separated by space). If any node does not have left or right child, take -1 in its place.

##### Output format :

Print nodes separated by new line.

##### Sample Input :

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

##### Sample Output :

9