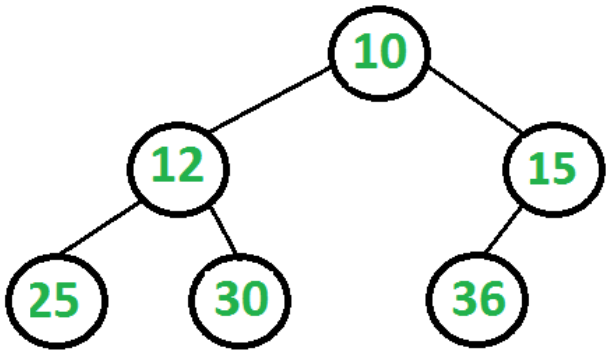
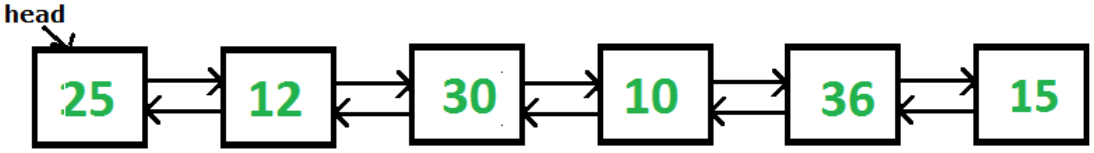
Binary Tree to DLL

Given a Binary Tree (BT), convert it to a Doubly Linked List (DLL) in place. The left and right pointers in nodes will be used as previous and next pointers respectively in converted DLL. The **order of nodes** in DLL must be the same as the order of the given Binary Tree. The first node of **Inorder traversal** (leftmost node in BT) must be the head node of the DLL.

Note: h is the tree's height, and this space is used implicitly for the recursion stack.



The above tree should be in-place converted to following Doubly Linked List(DLL).



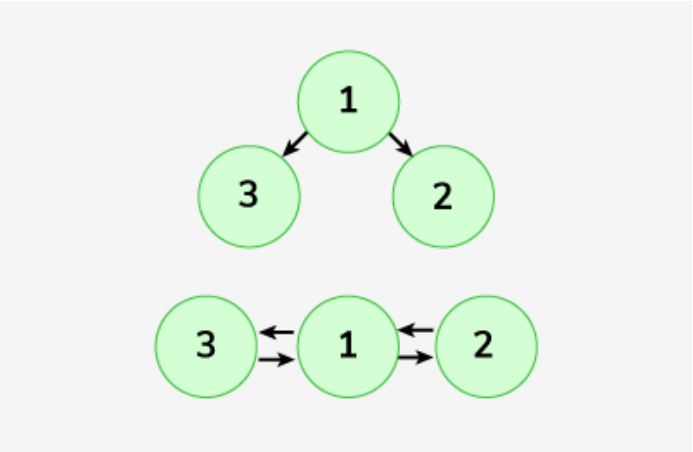
Examples:

Input:

```
    1
  /  \
 3    2
```

Output:
3 1 2

2 1 3



Explanation: DLL would be 3<=>1<=>2

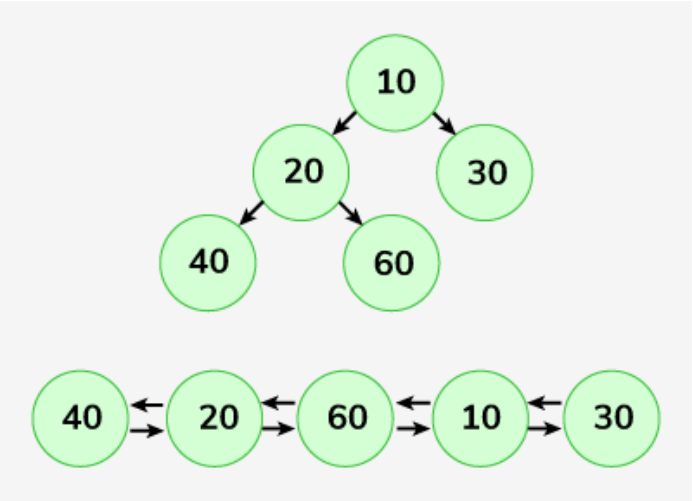
Input:

```
      10
     /   \
    20   30
  /   \
 40   60
```

Output:

40 20 60 10 30

30 10 60 20 40



Explanation: DLL would be 40<=>20<=>60<=>10<=>30.

Expected Time Complexity: O(no. of nodes)
Expected Space Complexity: O(height of the tree)

Constraints:
1 ≤ Number of nodes ≤ 10^5
0 ≤ Data of a node ≤ 10^5

```
class Node:
    """ Class Node """
    def __init__(self, value):
        self.left = None
        self.data = value
        self.right = None
'''

#Function to convert a binary tree to doubly linked list.
class Solution:
    def __init__(self):
        self.prev = None
        self.head = None

    def bToDLL(self, root):
        if root is None:
            return None

        self.bToDLL(root.left)

        if self.prev is None:
            self.head = root
        else:
            root.left = self.prev
            self.prev.right = root

        self.prev = root
        self.bToDLL(root.right)

        return self.head
```