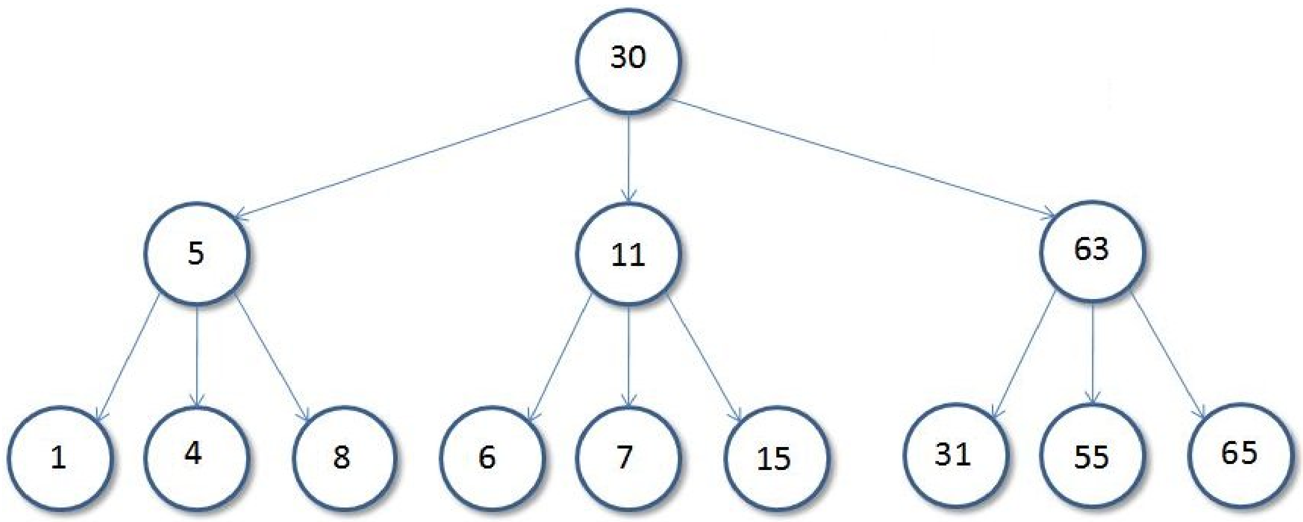
Create a Doubly Linked List from a Ternary Tree

Given a ternary tree, create a doubly linked list out of it. A ternary tree is just like binary tree but instead of having two nodes, it has three nodes i.e. left, middle, right.

The doubly linked list should holds following properties –

1. Left pointer of ternary tree should act as prev pointer of doubly linked list.
2. Middle pointer of ternary tree should not point to anything.
3. Right pointer of ternary tree should act as next pointer of doubly linked list.
4. Each node of ternary tree is inserted into doubly linked list before its subtrees and for any node, its left child will be inserted first, followed by mid and right child (if any).

For the above example, the linked list formed for below tree should be NULL <- 30 <-> 5 <-> 1 <-> 4 <-> 8 <-> 11 <-> 6 <-> 7 <-> 15 <-> 63 <-> 31 <-> 55 <-> 65 -> NULL



We strongly recommend you to minimize your browser and try this yourself first.

The idea is to traverse the tree in preoder fashion similar to binary tree preorder traversal. Here, when we visit a node, we will insert it into doubly linked list in the end using a tail pointer. That we use to maintain the required insertion order. We then recursively call for left child, middle child and right child in that order.

Below is C++ implementation of this idea.

```
// C++ program to create a doubly linked list out
// of given a ternary tree.
#include <bits/stdc++.h>
using namespace std;

/* A ternary tree */
struct Node
{
    int data;
    struct Node *left, *middle, *right;
};

/* Helper function that allocates a new node with the
   given data and assign NULL to left, middle and right
   pointers.*/
Node* newNode(int data)
{
    Node* node = new Node;
    node->data = data;
    node->left = node->middle = node->right = NULL;
    return node;
}

/* Utility function that constructs doubly linked list
by inserting current node at the end of the doubly
linked list by using a tail pointer */
void push(Node** tail_ref, Node* node)
{
    // initilize tail pointer
    if (*tail_ref == NULL)
    {
        *tail_ref = node;

        // set left, middle and right child to point
        // to NULL
        node->left = node->middle = node->right = NULL;

        return;
    }

    // insert node in the end using tail pointer
    (*tail_ref)->right = node;

    // set prev of node
    node->left = (*tail_ref);

    // set middle and right child to point to NULL
    node->right = node->middle = NULL;

    // now tail pointer will point to inserted node
    (*tail_ref) = node;
}

/* Create a doubly linked list out of given a ternary tree.
by traversing the tree in preoder fashion. */
Node* TernaryTreeToList(Node* root, Node** head_ref)
{
    // Base case
    if (root == NULL)
        return NULL;

    //create a static tail pointer
    static Node* tail = NULL;

    // store left, middle and right nodes
    // for future calls.
    Node* left = root->left;
    Node* middle = root->middle;
    Node* right = root->right;

    // set head of the doubly linked list
    // head will be root of the ternary tree
    if (*head_ref == NULL)
        *head_ref = root;

    // push current node in the end of DLL
    push(&tail, root);

    //recurse for left, middle and right child
    TernaryTreeToList(left, head_ref);
    TernaryTreeToList(middle, head_ref);
    TernaryTreeToList(right, head_ref);
}

// Utility function for printing double linked list.
void printList(Node* head)
{
    printf("Created Double Linked list is:\n");
    while (head)
    {
        printf("%d ", head->data);
        head = head->right;
    }
}

// Driver program to test above functions
int main()
{
    // Construting ternary tree as shown in above figure
    Node* root = newNode(30);

    root->left = newNode(5);
    root->middle = newNode(11);
    root->right = newNode(63);

    root->left->left = newNode(1);
    root->left->middle = newNode(4);
    root->left->right = newNode(8);

    root->middle->left = newNode(6);
    root->middle->middle = newNode(7);
    root->middle->right = newNode(15);

    root->right->left = newNode(31);
    root->right->middle = newNode(55);
    root->right->right = newNode(65);

    Node* head = NULL;

    TernaryTreeToList(root, &head);

    printList(head);

    return 0;
}
```

Output:

```
Created Double Linked list is:
30 5 1 4 8 11 6 7 15 63 31 55 65
```