Change a Binary Tree so that every node stores sum of all nodes in left subtree

Given a Binary Tree, change the value in each node to sum of all the values in the nodes in the left subtree including its own.

Example

```
Input :
     1
   /   \
 2      3

Output :
    3
  /   \
 2     3


Input
       1
      / \
     2   3
    / \   \
   4   5   6
Output:
      12
     / \
    6   3
   / \   \
  4   5   6
```

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

Source: http://qa.geeksforgeeks.org/1567/given-change-value-each-values-nodes-left-subtree-including

The idea is to traverse the given tree in bottom up manner. For every node, recursively compute sum of nodes in left and right subtrees. Add sum of nodes in left subtree to current node and return sum of nodes under current subtree.

Below is C++ implementation of above idea.

```
// C++ program to store  sum of nodes in left subtree in every
// node
#include<bits/stdc++.h>
using namespace std;

// A tree node
struct node
{
    int data;
    struct node* left,  *right;
};

// Function to modify a Binary Tree so that every node
// stores sum of values in its left child including its
// own value
int updatetree(node *root)
{
    // Base cases
    if (!root)
        return 0;
    if (root->left == NULL && root->right == NULL)
        return root->data;

    // Update left and right subtrees
    int leftsum  = updatetree(root->left);
    int rightsum = updatetree(root->right);

    // Add leftsum to current node
    root->data += leftsum;
    // Return sum of values under root
    return root->data + rightsum;
}

// Utility function to do inorder traversal
void inorder(struct node* node)
{
    if (node == NULL)
        return;
    inorder(node->left);
    printf("%d ", node->data);
    inorder(node->right);
}

// Utility function to create a new node
struct node* newNode(int data)
{
    struct node* node =
        (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}

// Driver program
int main()
{
    /* Let us construct below tree
                1
               / \
              2   3
             / \   \
            4   5   6    */
    struct node *root  = newNode(1);
    root->left         = newNode(2);
    root->right        = newNode(3);
    root->left->left   = newNode(4);
    root->left->right  = newNode(5);
    root->right->right = newNode(6);

    updatetree(root);

    cout << "Inorder traversal of the modified tree is \n";
    inorder(root);
    return 0;
}
```

Output:

```
Inorder traversal of the modified tree is
4 6 5 12 3 6
```

Time Complexity: O(n)