Double Tree

Write a program that converts a given tree to its Double tree. To create Double tree of the given tree, create a new duplicate for each node, and insert the duplicate as the left child of the original node.

So the tree...

```
    2
   / \
  1   3
```

is changed to...

```
       2
      / \
     2   3
    /   /
   1   3
  /
 1
```

And the tree

```
            1
          /   \
        2      3
      /  \
    4     5
```

is changed to

```
               1
             /   \
           1      3
          /      /
        2       3
      /  \
     2    5
    /    /
   4   5
  /
 4
```

Algorithm:
Recursively convert the tree to double tree in postorder fashion. For each node, first convert the left subtree of the node, then right subtree, finally create a duplicate node of the node and fix the left child of the node and left child of left child.

Implementation:

C

```
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
```

```
};

/* function to create a new node of tree and returns pointer */
struct node* newNode(int data);

/* Function to convert a tree to double tree */
void doubleTree(struct node* node)
{
  struct node* oldLeft;

  if (node==NULL) return;

  /* do the subtrees */
  doubleTree(node->left);
  doubleTree(node->right);

  /* duplicate this node to its left */
  oldLeft = node->left;
  node->left = newNode(node->data);
  node->left->left = oldLeft;
}



/* UTILITY FUNCTIONS TO TEST doubleTree() FUNCTION */
 /* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
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);
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
{
  if (node == NULL)
    return;
  printInorder(node->left);
  printf("%d ", node->data);
  printInorder(node->right);
}


/* Driver program to test above functions*/
int main()
{

  /* Constructed binary tree is
            1
          /   \
        2      3
      /  \
    4     5
  */
  struct node *root = newNode(1);
  root->left        = newNode(2);
  root->right       = newNode(3);
  root->left->left  = newNode(4);
  root->left->right = newNode(5);

  printf("Inorder traversal of the original tree is \n");
  printInorder(root);

  doubleTree(root);

  printf("\n Inorder traversal of the double tree is \n");
  printInorder(root);
  printInorder(root);

  getchar();
  return 0;
}
```

Java

```
// Java program to convert binary tree to double tree

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
class Node
{
    int data;
    Node left, right;

    Node(int item)
    {
        data = item;
        left = right = null;
    }
}

class BinaryTree
{
    Node root;

    /* Function to convert a tree to double tree */
    void doubleTree(Node node)
    {
        Node oldleft;

        if (node == null)
            return;

        /* do the subtrees */
        doubleTree(node.left);
        doubleTree(node.right);

        /* duplicate this node to its left */
        oldleft = node.left;
        node.left = new Node(node.data);
        node.left.left = oldleft;
    }

    /* Given a binary tree, print its nodes in inorder*/
    void printInorder(Node node)
    {
        if (node == null)
            return;
        printInorder(node.left);
        System.out.print(node.data + " ");
        printInorder(node.right);
    }

    /* Driver program to test the above functions */
    public static void main(String args[])
    {
        /* Constructed binary tree is
              1
            /   \
           2     3
         /  \
        4    5
        */
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);

        System.out.println("Original tree is : ");
        tree.printInorder(tree.root);
        tree.doubleTree(tree.root);
        System.out.println("");
        System.out.println("Inorder traversal of double tree is : ");
        tree.printInorder(tree.root);
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

Time Complexity: O(n) where n is the number of nodes in the tree.

References:
http://cslibrary.stanford.edu/110/BinaryTrees.html