Check Identical Trees

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

struct node
{
	int data;
	struct node *left;
	struct node *right;
};

struct node *newNode(int data)
{
	struct node *temp = (struct node *)malloc(sizeof(struct node));
	temp->data = data;
	temp->left = temp->right = NULL;
	return temp;
}

int checkIdenticalTrees(struct node *root1, struct node *root2)
{
	if(!root1 && !root2)
		return 1;
	if(root1 && root2)
	{
		return (root1->data == root2->data)&&
		checkIdenticalTrees(root1->left, root2->left)&&
		checkIdenticalTrees(root1->right, root2->right);
	}
	return 0;
}

int main()
{
	struct node *root1, *root2;
	root1 = newNode(10);
	root1->left = newNode(20);
	root1->right = newNode(30);
	root1->left->left = newNode(40);
	root1->right->left = newNode(50);
	root2 = newNode(10);
	root2->left = newNode(20);
	root2->right = newNode(30);
	root2->left->left = newNode(40);
	root2->right->left = newNode(50);
	checkIdenticalTrees(root1, root2)?printf("Both are identical"): printf("Both are not
identical\n");
	return 0;
}
```

Time complexity: O(n)
Space Complexity: O(n)