Mirror Tree

```
#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;
}

void swap(struct node **left, struct node **right)
{
	struct node *temp = *left;
	*left = *right;
	*right = temp;
}

void getMirrorTree(struct node *root)
{
	if(root)
	{
		getMirrorTree(root->left);
		getMirrorTree(root->right);
		swap(&root->left, &root->right);
	}
}

void inorder(struct node *root)
{
	if(root)
	{
		inorder(root->left);
		printf("%d\t", root->data);
		inorder(root->right);
	}
}

int main()
{
	struct node *root=NULL;
	root = newNode(10);
	root->left = newNode(20);
```

```
        root->right = newNode(30);
        root->left->left = newNode(40);
        root->right->left = newNode(50);
        getMirrorTree(root);
        inorder(root);
        return 0;
}
```

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