```c
// C program to convert a binary tree to its mirror
#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;
};

/* 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);
}


/* Change a tree so that the roles of the left and
    right pointers are swapped at every node.

So the tree...
    4
    / \
    2 5
    / \
1 3

is changed to...
```

```
40:     4
41:     / \
42:     5 2
43:         / \
44:     3 1
45: */
46: void mirror(struct Node* node)
47: {
48: if (node==NULL)
49:     return;
50: else
51: {
52:     struct Node* temp;
53:
54:     /* do the subtrees */
55:     mirror(node->left);
56:     mirror(node->right);
57:
58:     /* swap the pointers in this node */
59:     temp      = node->left;
60:     node->left = node->right;
61:     node->right = temp;
62: }
63: }
64:
65: /* Helper function to print Inorder traversal.*/
66: void inOrder(struct Node* node)
67: {
68: if (node == NULL)
69:     return;
70:
71: inOrder(node->left);
72: printf("%d ", node->data);
73: inOrder(node->right);
74: }
75:
76:
77: /* Driver program to test mirror() */
78: int main()
```

```
 79: {
 80: struct Node *root = newNode(1);
 81: root->left   = newNode(2);
 82: root->right  = newNode(3);
 83: root->left->left = newNode(4);
 84: root->left->right = newNode(5);
 85:
 86: /* Print inorder traversal of the input tree */
 87: printf("Inorder traversal of the constructed"
 88:         " tree is \n");
 89: inOrder(root);
 90:
 91: /* Convert tree to its mirror */
 92: mirror(root);
 93:
 94: /* Print inorder traversal of the mirror tree */
 95: printf("\nInorder traversal of the mirror tree"
 96:         " is \n");
 97: inOrder(root);
 98:
 99: return 0;
100: }
101:
```