def UNMIRROR(x):

Step 0: if x is None, return

Step 1: x.right, x.left = x.left, x.right

Step 2: UNMIRROR(x.left)

Step 3: UNMIRROR(x.right)

Step 4: return x

Correctness:

Think about any two nodes (A,B) such that A is left to B in the original tree.

Case 1: A is in the left subtree of B. After UNMIRROR(B) is called, left subtree of B becomes the right subtree of A, so now A is right to B

Case 2: B is in the right subtree of A. Similar to Case 1, after running B is in the left subtree of A. A is right to B

Case 3: A and B shares an ascendant. Suppose C is the "lowest" ascendant, A is in the left subtree of C and B is in the right subtree of C. After UNMIRROR(C) is called, A is in the right subtree of C and B is in the left subtree of C. A is right to B.

So for any two nodes in the original subtree, the order of the two will be swapped. The original tree is reversed.

Time complexity:

The function UNMIRROR is called exactly at most 3n times consider non-existent children, and it takes constant times to execute UNMIRROR every time, the total run time is O(n)