When we swap a node to a leaf, do we find the leaf node to swap with first, or swap the node with one of its children all the way down the tree until the node becomes a leaf?

Consider a binary search tree built from the following input:

C A B

If we delete C, the root, does A become the new root or does B?

If we find the leaf to swap with first, then swap and prune, B becomes the root.

If we swap C with A, then B with C, then A becomes the root.

Both would maintain correct BST properties, but yield separate results.

Here is my pseudocode, which yields different results from David McCoy's outputs in his testing thread.

//edit: wrong, see sample output below

```cpp
swapToLeaf(node){
    if (node has a left child)
        swap with the left child
    while (node is not a leaf)
        if (node has a right child) swap with it
        else swap with the left child
    if (node has a right child)
        swap with the right child
    while (node is not a leaf)
        if (node has a left child) swap with it
        else swap with the right child
}
```

David's pseudocode for comparison:

```cpp
swapToLeaf(node)
    if (node is leaf) return node
    bstnode np = NULL
    if (node has a left child) np = predecessor(node)
    else np = successor(node)
    swap node's value with np's value
```
return swapToLeaf(np)

After inserting G H A E C B D into a tree and then deleting G, my output is:
//edit: duh, e can't be the left child of a

0: "a"("a")-
1: "e"("a")-l="h"("a")-r
2: "c"("e")-r
3: "d"("c")-l
4: ="b"("d")-l

David's is:

0: "e"("e")-
1: "a"("e")-l="h"("e")-r
2: "d"("a")-r
3: "c"("d")-l
4: ="b"("c")-l