```cpp
// Connor Petri
// CIS 22C
// 2023-11-7

/*
 * LHC = Left hand child; RHC = Right hand child
 *
 * If current node has a LHC, go to the it and walk down RHC until a leaf node
 * is encountered
 * Else if current node is the RHC of parent, return parent
 * Else move up the tree until current node is a RHC of parent and return it's parent
 * If root is encountered, then there is no previous node and you are at the tree's
 * minimum.
 */

Node * getPrevious(Node *from)
{
    Node *n = from; // current node

    if (n->LHC) // If LHC exists
    {
        n = n->LHC; // Go to LHC
        while (n->RHC) // Walk down RHC until leaf node encountered
        {
            n = n->RHC;
        }
        return n; // return encountered leaf
    }

    if (n->PARENT->RHC == n) // Else if node is the RHC of parent, return parent
    {
        return n->PARENT;
    }

    while (n->PARENT->RHC != n) // loop until RHC found (or root encountered)
    {
        if (!n->PARENT) // if root encountered, return null as there is no previous
        { return nullptr; }

        n = n->PARENT; // walk up the tree one node
    }
    return n; // n only returns here if a RHC is found and the root node is not
encountered
}
```