/// Andrew Souza

/// Comp 210 -- Spring 2024

/// Final Exam

#include <iostream>

#include "IntBinaryTree-3.h"

using namespace std;

// In the BinarySearchTree class, complete the makeDeletion member function

// so that a node with two children is replaced by the largest child of the

// left or right subtree. You may choose.

void IntBinaryTree::makeDeletion(TreeNode \*&nodePtr)

{

// Define a temporary pointer to use in reattaching

// the left subtree.

TreeNode \*tempNodePtr = nullptr;

if (nodePtr == nullptr)

cout << "Cannot delete empty node.\n";

else if (nodePtr->right == nullptr)

{

tempNodePtr = nodePtr;

nodePtr = nodePtr->left; // Reattach the left child

delete tempNodePtr;

}

else if (nodePtr->left == nullptr)

{

tempNodePtr = nodePtr;

nodePtr = nodePtr->right; // Reattach the right child

delete tempNodePtr;

}

// If the node has two children.

else

{

tempNodePtr = nodePtr->right;

while (nodePtr->left) {

nodePtr = nodePtr->left;

}

if (tempNodePtr->left > nodePtr) {

nodePtr->right = tempNodePtr->left;

} else {

nodePtr->left = tempNodePtr->left;

}

delete tempNodePtr;

}

}

int main() {

IntBinaryTree myTree;

myTree.insertNode(5);

myTree.insertNode(3);

myTree.insertNode(8);

myTree.insertNode(7);

myTree.insertNode(6);

myTree.insertNode(4);

myTree.insertNode(1);

myTree.insertNode(9);

myTree.insertNode(2);

cout << "Original:\n";

myTree.displayPreOrder();

myTree.remove(3);

cout << "After Removal:\n";

myTree.displayPreOrder();

return 0;

}

