CSC 122 001 Computer Science II

Julius Ranoa

Documentation for Two-Child Tree Node Removal from a Binary Tree.

The following documentation is for this snippet:

```
// Move to right subtree.
attachPoint = tree->right;

// Locate the smallest node in the right subtree
// by moving as far to the left as possible.
while (attachPoint->left != nullptr)
    attachPoint = attachPoint->left;

// Attach the left subtree of the original tree
// as the left subtree of the smallest node
// in the right subtree.
attachPoint->left = tree->left;

// Replace the original tree with its right subtree.
tree = tree->right;
```

For the full source, see class IntBinaryTree *implementation from text.*

The removal of tree node with two children from a binary tree is done as follows:

1. The pointer, *attachPoint*, finds the leftmost node of the right subtree of the node to be deleted.
2. The left subtree of the node to be deleted is attached to the leftmost node of right subtree.
3. The node to be deleted is replaced by its right subtree.
4. The node to be deleted is deleted from memory.

The following code is given to see this in action:

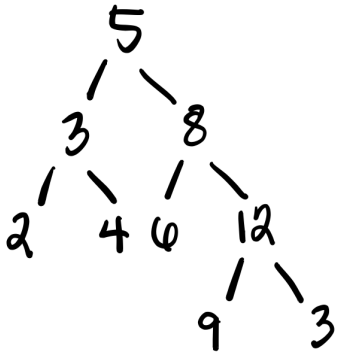
```
#include <iostream>
#include "IntBinaryTree.h"
using namespace std;

int main()
{
    IntBinaryTree tree;

    cout << "Inserting the numbers 5 8 3 12 9.\n";
    tree.insert(5);
    tree.insert(8);
    tree.insert(3);
    tree.insert(12);
    tree.insert(9);
    cout << "Inserting additional numbers, 2 4 6 13, to create two-children nodes.\n";
    // Adding the following numbers to create nodes with two children.
    // Nodes with two children are: 5, 3, 8, 12.
```

```
    tree.insert(2);
    tree.insert(4);
    tree.insert(6);
    tree.insert(13);

    cout << "\nHere are the values in the tree:\n";
    tree.showInOrder();
    cout << "\nHere are the values in pre-order: \n";
    tree.showPreOrder();

    cout << "\n\nThe delete method has been altered to show \n";
    cout << "that the method is deleting a node with two children. \n";

    cout << "\nDeleting 8...\n";
    tree.remove(8);

    cout << "\nDeleting 12...\n";
    tree.remove(12);

    cout << "\nDeleting 3... \n";
    tree.remove(3);

    cout << "\nNow, here are the nodes:\n";
    tree.showInOrder();
    cout << "\nNow, here are the values in pre-order: \n";
    tree.showPreOrder();
    return 0;
}
```

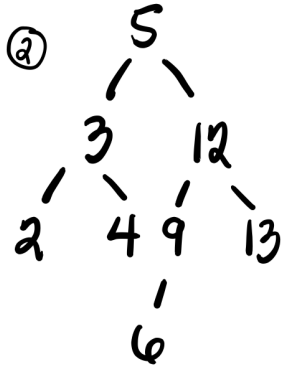
The screenshot of runtime is as follows:

```
Inserting the numbers 5 8 3 12 9.
Inserting additional numbers, 2 4 6 13, to create two-children nodes.

Here are the values in the tree:
2  3  4  5  6  8  9  12  13
Here are the values in pre-order:
5  3  2  4  8  6  12  9  13

The delete method has been altered to show
that the method is deleting a node with two children.

Deleting 8...
Deleting a node with two children...

Deleting 12...
Deleting a node with two children...

Deleting 3...
Deleting a node with two children...

Now, here are the nodes:
2  4  5  6  9  13
Now, here are the values in pre-order:
5  4  2  13  9  6
```

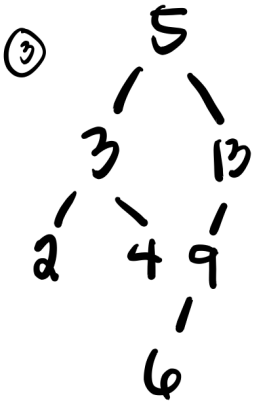
The binary tree after all the insertions should have the following structure:



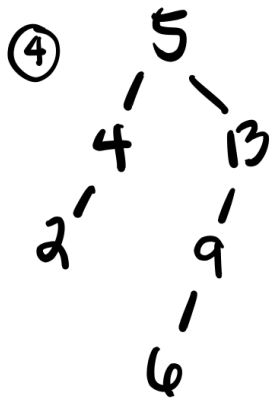
After the first removal of the node 8, the right subtree with node 12 at the head gets promoted and the left subtree with node 6 is attached to node 9.
The new structure is as follows:



After the second removal (node 12), node 13 gets promoted and the left subtree with node 9 at the head gets attached as the left subtree of node 13.
The new structure is as follows:



After the last removal (node 3), node 4 gets promoted and node 2 gets attached as the left subtree of node 4.
The new structure is as follows:



This matches the pre-order display of the tree after the last three removals.