CSCI-2270
Recitation 8
Instructor: Boese

Deleting a node in a BST

Objectives

- Delete a node in a BST

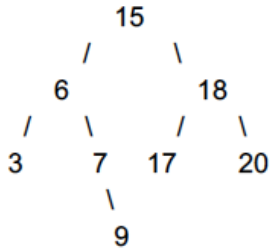
In this recitation, you will be learning how to delete a node in binary search trees.

Deleting a Node in Binary Trees

When a node is deleted from the tree, the node may need to be replaced with another node in the tree. The replacement node needs to be selected such that the BST properties are preserved. There are three cases to consider when deleting a node. Exactly one of the following conditions is true about the deleted node:

1. The node has no children.
2. The node has one child.
3. The node has two children.

The tree below shows a BST with examples of nodes with 0, 1, or 2 children. The nodes with values of 3, 9, 17, and 20 have no children. The node with a value of 7 has one child, and the nodes with values of 6, 15, and 18 have two children. The `delete()` algorithm to handle all three cases is below. For brevity, only the case where the deleted node is its parent's left child is shown.



Algorithm delete(value)

Deletes the node where the value matches the node key value.

Pre-conditions

value is a valid search value whose type matches the node key type.
search() algorithm exists to identify the node to delete.
treeMinimum() algorithm exists to identify the minimum value in a sub-tree, which will be the replacement node for a deleted node with two children.

Post-conditions

Node with specified key value is deleted from the tree.
parent, left child, and right child pointers for the deleted node and neighboring nodes are reset accordingly.

Algorithm

(Note: this is not the complete delete() algorithm. For the one- and two-children cases, only the case where the deleted node is the left child of its parent is shown. Additional cases are needed to handle when the deleted node is the right child.)

```
delete(value)
1.  node = search(value)
2.  if(node != root)
3.      if(node.leftChild == NULL and node.rightChild == NULL)        //no children
4.          node.parent.leftChild = NULL
5.      else if(node.leftChild != NULL and node.rightChild != NULL)   //two children
6.          min = treeMinimum(node.rightChild)
7.      if (min == node.rightChild)
8.          node.parent.leftChild = min
10.     else
11.         min.parent.leftChild = min.rightChild
12.         min.parent = node.parent
13.         min.right.parent = min.parent
14.         node.parent.leftChild = min
15.         min.leftChild = node.leftChild
16.         min.rightChild = node.rightChild
17.         node.rightChild.parent = min
18.         node.leftChild.parent = min
19.     else                                                          //one child
20.         x = node.leftChild
21.         node.parent.leftChild = x
22.         x.parent = node.parent
23.     else
24.         //repeat cases of 0, 1, or 2 children
25.         //replacement node is the new root
26.         //parent of replacement is NULL
27. delete node
```

Examples to cover:

Node has no children

delete(3)

Steps:

1. Set the left child pointer for the 6 to NULL.
2. Delete the 3 node.

Node has one child

delete(7)

Steps:

1. Set the right child pointer of the 6 to point to the 9.
2. Delete the 7 node.

Node has two children

delete(6)

Steps:

1. The parent property of the 3 is updated to point to the 7.
2. The leftChild property of the 15 is updated to point to the 7.
3. The parent property of the 7 is updated to point to the 15.
4. Delete the 6 node.

Recitation Assignment

PART 1

Write a C++ function to delete the given value from the binary search tree. The function takes two arguments: the tree node and value of the node to be deleted.

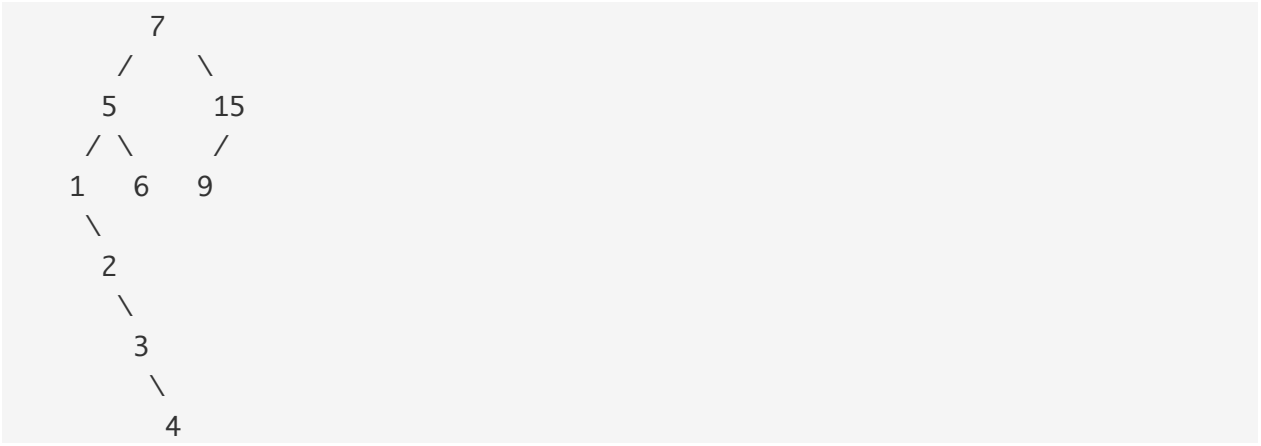
Also replace the node to be deleted with maximum value from left sub tree when the node has two children.

Note: this is not the same delete algorithm expected for the Assignment.

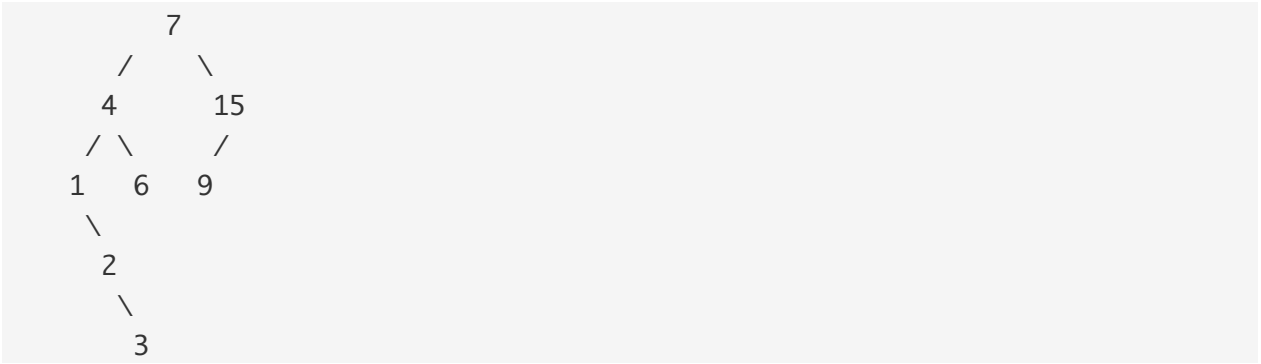
```
void deleteNode(TreeNode *node, int key);
```

```
struct TreeNode
{
      int key;
      TreeNode *left;
      TreeNode *right;
      TreeNode *parent;
};
```

For example:



If 5 is deleted, then replace it with the maximum value in its left subtree which is 4. The final tree:



PART 2

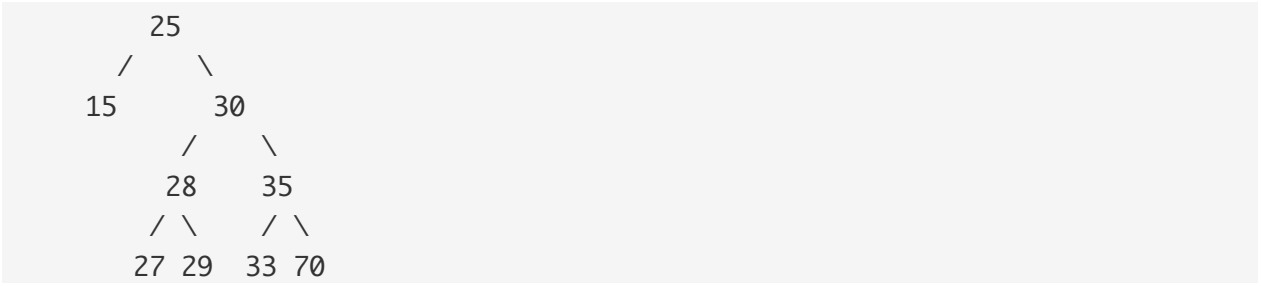
Write a C++ function to delete the given value from the binary search tree. The function takes two arguments, tree node and value of the node to be deleted.

Also replace the node to be deleted with the minimum of its right subtree when the node has two children.

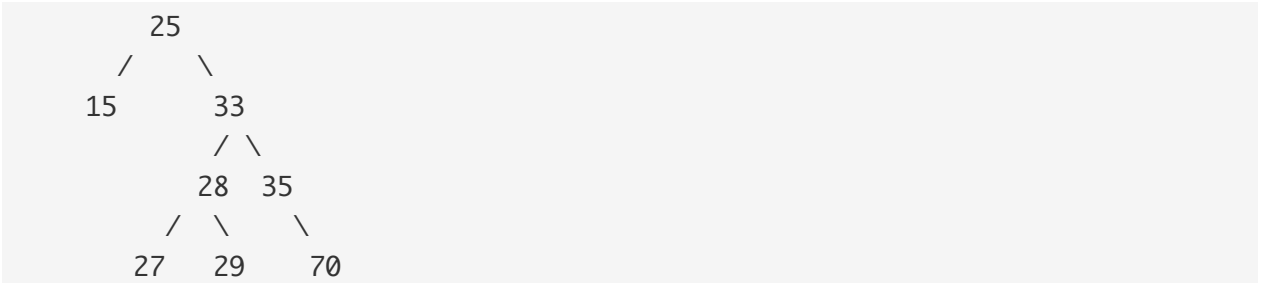
```
void deleteAndReplaceMinRight(TreeNode *root, int key);
```

```
struct TreeNode
{
      int key;
      TreeNode *left;
      TreeNode *right;
      TreeNode *parent;
};
```

For example:



If the node to be deleted is 30, delete it and replace it with the minimum of its right subtree. Final tree:



To get credit for this lab exercise, show the TA and sign the lab's completion log.