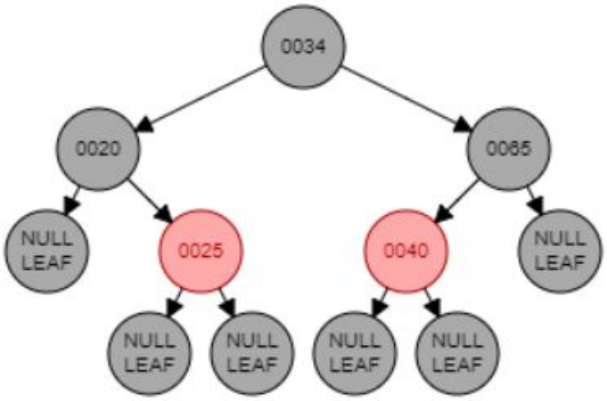
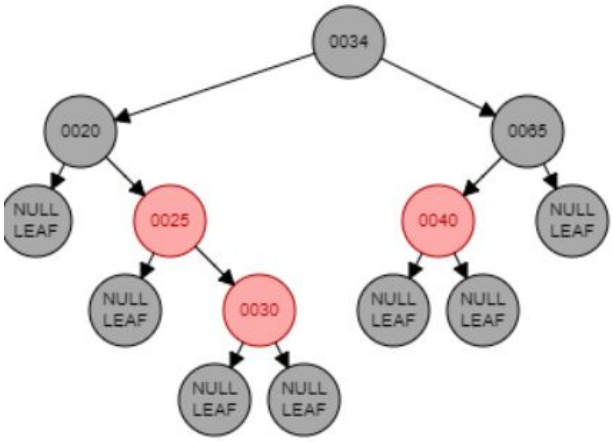
Assignment 7
Ryan Hoffman
CSCI 2270
TA: Abhidip Bhattacharyya

Question 1) Does inserting a node into a red-black tree, re-balancing, and then deleting it result in the original tree?

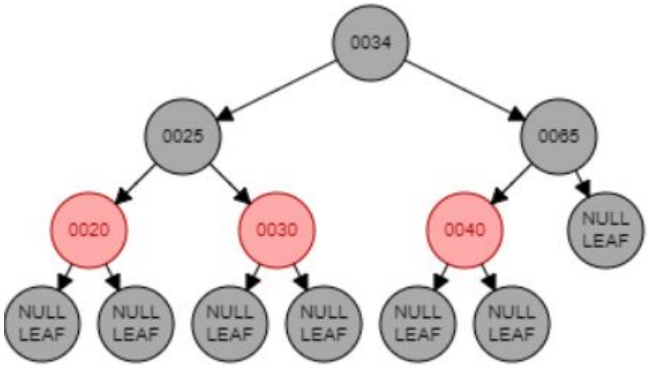
Answer:
Let's use the following tree:



First we will insert a node.
insert(0030)
0030<0034 //Looking at left subtree
0030>0020 //Looking at right subtree
0030>0025 //Looking at right subtree
0030 is now right child of 0025



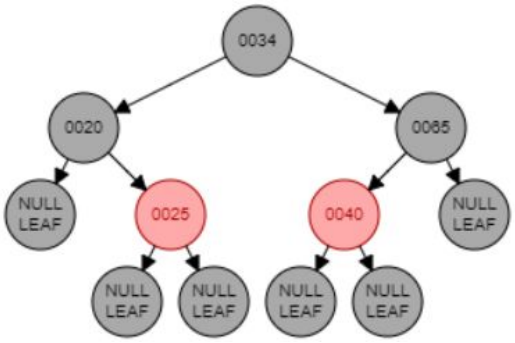
Next, since node and parent are both red and node is right child and parent is right child, we need to rotate left and re-color 0025 to black. **Rebalance()**



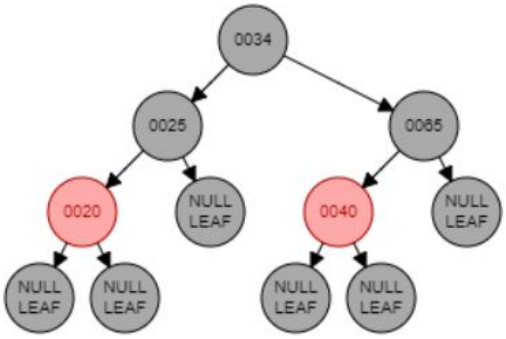
Now, we will delete the value, 0030.
delete(30)
0030<0034 //Looking at left subtree.
0030>0025 //Looking at right subtree.
0030==0030 //Found node to delete.
0030 is a leaf node so we just delete it.

The resulting tree is NOT the original tree as we can see from comparing the two images:

BEFORE:



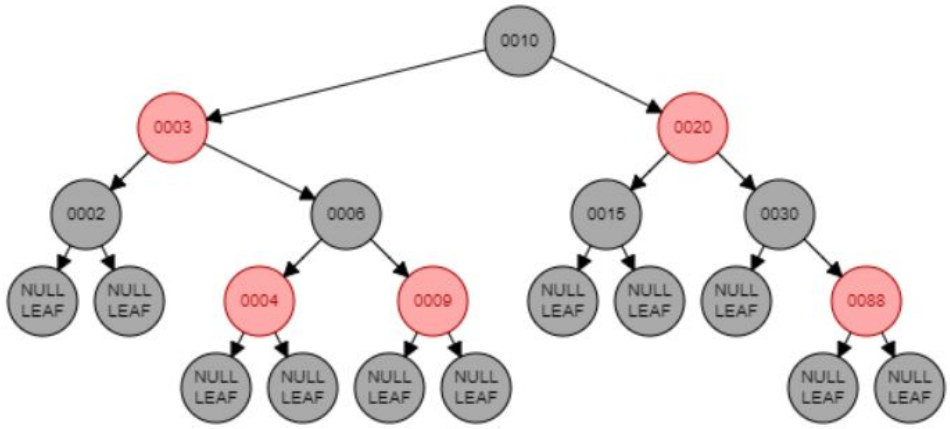
AFTER:



End of Question 1.

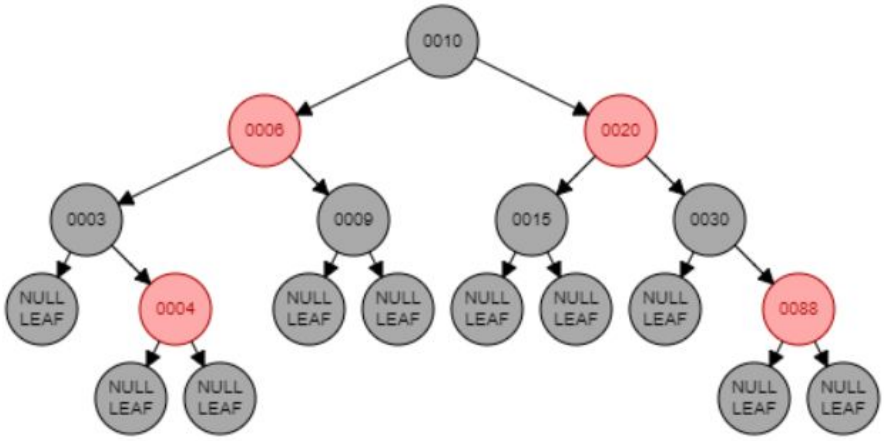
Question 2)Does deleting a node with no children from a red-black tree, rebalancing, and then reinserting it with the same key always result in the original tree?

Answer:
Let's use the following tree:



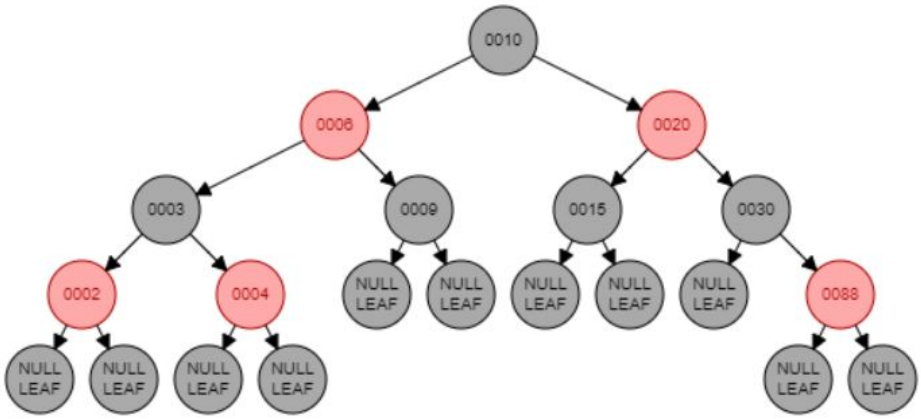
First, we will delete the value, 0002.
delete(2)
0002<0010 //Looking at the left subtree.
0002<0003 //Looking at the left subtree.
0002==0002 //Found node to delete.
Node is a leaf node so we can just delete it.

Now the right nephew of the left child of 0003 is red so we need to rotate left.
rebalance() gives us the following tree:



Finally we will insert the value, 0002 back in to the tree:
insert(2)
0002<0010 //Looking at left subtree
0002<0006 //Looking at left subtree
0002<0003 //Looking at left subtree
0002 is now left child of 0003.

Here is the tree now:



The resulting tree is NOT the original tree as we can see from comparison.

End of Question 2.