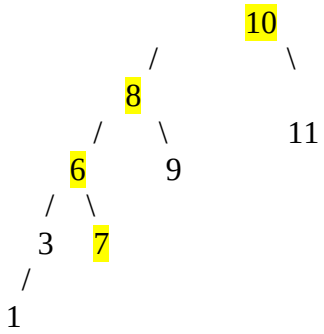
COP 3502 Suggested Program Edits: Binary Search Tree Delete, AVL Trees (Week 9 Programs)

1) Write a function that takes in the root of a binary tree and returns the sum of the path going down the tree that stores the maximal sum. For example, for the tree below:



the answer is 31 for the nodes highlighted in yellow. Note: Even though this tree is a search tree, your code should work for any binary tree, not just binary search trees.

2) Add debugging print statements to the recursive delete code in bintree.c to help elucidate the order in which things occur.

3) Write your own binary search tree function that always either branches left or right (so one recursive call only) based on some property of the value stored in a node, and produces some output. Then, create a tree and see if you can trace through your code to generate the correct output. Run your code to check if your trace was correct!

4) In class we worked out the worst case height for a binary search tree of n nodes. (Specifically, if $n \leq F_{h+3} - 1$, then the height of the tree is no more than h, with equality occurring potentially if $n = F_{h+3} - 1$. Thus, given any value n, and the Fibonacci sequence, we could work out the theoretical worst case height for an AVL tree. Write a test that inserts a million random items into an AVL tree and checks to see how close the height of the tree is to the worst possible case.

5) The file avltree.c finds the maximum node on the left when asked to delete a node with two children. The alternate strategy that works is finding the minimum node on the right. Edit avltree.c to use this strategy instead. Make sure all of the previous tests still work.