CS201 - Assignment 9, Part 1
Due: Friday, April 25, *at the beginning of class*

http://pj.sushovan.de/page/3

For part 1 of this assignment you may (and I'd encourage you to) work with a partner. **For part 2, however, it will be a solo assignment.**

For this assignment, you will only be handing in a .txt file with some answers, though it will involve some coding.

How many trees can you make?

Question 1: How many binary search trees can you make with the numbers 1 through 4?

How tall do trees really get?

Question 2: If you insert 127 numbers into a binary search tree, what is the tallest the tree can get?

Question 3: If you insert 127 numbers into a binary search tree, what is the shortest the tree can get?

Finally, let's investigate how tall the tree gets on average when you insert 127 random numbers. At:

`http://www.cs.middlebury.edu/~dkauchak/classes/cs201/assignments/assign9/part1/`

I've included an implementation of a binary search tree that utilizes our `BinaryTree` implementation.

Write some code in another class that creates a new `BinarySearchTree` and inserts 127 random numbers and then measures the height of the binary search tree (recall the `java.util.Random` class is useful for generating random numbers). Put this into a loop and repeat it 100 times. Average the heights.

Question 4: If you insert 127 numbers into a binary search tree, what is the average height of the tree? How does this compare to the optimal value?

Question 5: If you insert 1023 numbers into a binary search tree, what is the average height of the tree? First, see if you can guess the height (using math :), then measure it empirically.

When you're done

Make sure your name(s) is at the the of your file.

Submit the .txt file with the answers to your questions through the normal submission mechanism as "9.1". No need to submit any code (unless you do the extra credit).

Extra Credit

What I've provided is a very stripped down version of a binary search tree. Add a `delete` method that takes a value as a parameter, searches through the tree to find that value and, if it exists, deletes it from the tree. I'd recommend first implementing the `successor` method, which you'll need for deleting a node with two children, before implementing `delete`.