I've started out by writing my tree node class and my bst class, which took me a few minutes since I adapted my node and bst classes from calculon. I added a delete routine. You should omit the 'splice out' case because AVL*. Remember to use strcmp for comparing keys. I need to test insertion, finding, and deletion thoroughly before continuing. I'll write a random command generator and some tree checking routines (balance and ordering), as well.

*doing some hip interweb speak, here.

Before starting on a complicated data structure, the first thing you should do is write some support code that will test your future data structure. Then, as soon as you get some aspect of your data structure working, you can test it. I've uploaded an AVL tree checker, but have deleted some parts that you will need to implement, mainly a queue for holding the checker's version of AVL tree nodes and a BST ordering test. You can get and install the checker with these commands:

```
cd
mkdir avlchecker
cd avlchecker
wget beastie.cs.ua.edu/cs201/avlchecker.tgz
tar xvfz avlchecker.tgz
make
./avlchecker avl.1
```

This last command checks a tree found in avl.1. It won't work, though, until you edit queue.c and finish the queue. Once you've done that, avlchecker should give a pass to the tree in avl.1 with regards to AVL balance. It will then fail the ordering check since that's not finished either.