Here are some problems that I assigned the last time I taught CSSE 230. They can be good practice/review problems before you begin 473. Some of them may actually become 473 assignments.

Many of the problems are from Mark Weiss's Data Structures and Problem Solving Using Java. If I list a number in parentheses after a problem number (such as 3.4), that refers to a problem 4 at the end of chapter 3 of Weiss. Two numbers (3.5/3.7) means that it is problem 3.5 in the 3rd edition of Weiss (hardcover) and 3.7 in the 4th edition (paperback).

These problems are indicative of the kind of background that I hope students coming into 473 will have. We will of course review and/or extend some of these concepts in the course

1. (8.10) Using Stirling's formula, $N! \geq (N/e)^N$ sqrt(2πN) , derive an estimate for log(N!)

2. (19.2) a. Draw all Binary Search Trees (BSTs) that can result from starting with an empty tree and inserting the numbers 1, 2, 3, and 4 in some order. How many different trees are there. If all insertion orders are equally likely, what is the probability of each tree occurring? [Recall that in a BST, all elements in the left subtree have values that are smaller than the root, and all elements i the right subtree have values that are larger than the root. This is also true for each nonempty subtree]

   b. Do the same thing, but with the restriction that the BST is an AVL tree.

3. (18.8) Suppose that a binary tree has leaves $l_1, l_2, \ldots l_M$, whose depths are $d_1, d_2, \ldots d_M$, respectively. Prove by induction that $\sum_{i=1}^{M} 2^{-d_i} \leq 1$. For some trees, the inequality can be replaced by an equals sigh. Which trees are those?

   In your proof by (strong) induction, use our standard recursive definition of a binary tree --a binary tree is either empty or it a root and two subtrees, both subtrees are (possibly empty) binary trees -- as the basis for your induction. Notice that the formula involves the *depths* of the nodes, not their heights. For example, in the tree in Figure 18.33, the depth of node J is 3, while the depth of C is 1. As always, be sure to clearly state the induction hypothesis and indicate which step(s) use that hypothesis.

   For the second part of the problem, I am looking for a (brief) general description of the set of binary trees for which we can replace the $\leq$ in the given formula by =.

4. (20 points) Fill in the following table. Be very careful to get the exact values.
   Half of the credit will be for the last column. Feel free to include explanations of your answers.
   (wrong answer) + (no explanation) = (no credit). Recall that the height of a 1-node tree is 0.

N	**Height of shortest binary tree with N nodes**	**Height of tallest binary tree with N nodes**	**Height of tallest AVL tree with N nodes**
7	**2**	**6**	**3**
8			
370			
17000			
50000			

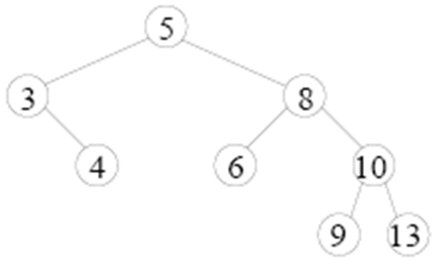
5. Use the recursive definition of "binary tree" as the basis of a proof by (strong) induction of the following statement. Be very explicit about what is the base case, what is the induction assumption, what you are trying to show in the induction step, and where you use the induction assumption when

you show it.

If T is a **nonempty** binary tree, let F(T) be the number of full nodes (nodes with two children) of T, and let L(T) be the number of leaves (nodes with no children) of T. [Note that T may also contain nodes that have exactly one child, but they do not contribute to either F(T) or L(T).]

Show that for every nonempty binary tree T, L(T) = F(T) + 1.

6. Start with the following Binary Search Tree:



(a) Is this an AVL tree? ____ If not, rearrange it so that it is height-balanced.

(b) Draw the tree after insertion of a node containing 11, using the usual BST insertion algorithm.

(c) Is the new tree AVL? ______ If not, name the node where the rotation should be done, according to the algorithm from class. ______ Single or double rotation? ____________ If you need to do a rotation, draw the resulting tree.

(d) Delete the element 5 from the original tree, using the algorithm described in class and in the textbook. Draw the new tree.

(e) Is the new tree AVL? ______ If not, name the node where the rotation should be done, according to the algorithm from class. ______ Single or double rotation? ____________ If you need to do a rotation, draw the resulting tree.

(f) Is your new tree AVL? ______ If not, name the node where the rotation should be done, according to the algorithm from class. ______ Single or double rotation? ____________ If you need to do a rotation, draw the resulting tree.

(g) Add the element 5 to the tree from part (f). Draw the new tree.

(h) Is the new tree AVL? ______ If not, name the node where the rotation should be done, according to the algorithm from class. ______ Single or double rotation? ____________ If you need to do a rotation, draw the resulting tree.

(i) Add the element 12 to the tree from part (h). Draw the new tree.

(j) Is the new tree AVL? ______ If not, name the node where the rotation should be done, according to the algorithm from class. ______ Single or double rotation? ____________ If you need to do a rotation, draw the resulting tree.

(k) Add the element 7 to the tree from part (j). Draw the new tree.

(l) Is the new tree AVL? ______ If not, name the node where the rotation should be done, according to the algorithm from class. ______ Single or double rotation? ____________ If you need to do a rotation, draw the resulting tree.

7. (19.8) Show the result of starting with an initially empty AVL tree and inserting the numbers 1 through 15 in that order. Generalize the result (and prove it by induction) to the case where the numbers 1 through 2^k-1 are inserted into an initially empty AVL tree in order. Prove your result by induction. Don't expect to do this one in an hour. The proof, while conceptually not too hard, takes some serious thought (and probably a lot of words and diagrams) to figure out how to express it. Also, what you want to prove in the end may be a special case of a more general thing that you will prove by induction. Don't gloss over too many details. Convince me that you really know why it is true and that you can organize the induction proof well. My write-up of this proof takes a couple of pages.

8. In this problem, we consider completely full binary trees with N nodes and height H (so that $N = 2^{H+1} - 1$)

   (a) (5 points) Show that the sum of the heights of all of the nodes of such a tree can be expressed as

$$\sum_{k=0}^{H} k\,2^{H-k} \quad .$$

   (b) (15 points) Prove by induction on H that the above sum of the heights of the nodes is N - H - 1. You may base your proof on the summation from part (a) (so you don't need to refer to trees at all), **or** you may do our "standard" binary tree induction based on the heights of the trees, using the definition that a non-empty binary tree has a root plus left and right subtrees. I find the tree approach more straightforward, but you may use the summation if you prefer.

9. Use iteration to solve the following recurrence relations:

$a_1 = 1$. $a_n = 2a_{n-1} + 1$

$a_0 = 1$. $a_n = 2na_{n-1}$

10. Suppose we exchange elements a[i] and a[i+k] Show that at least 1 and at most 2k-1 inversions are removed by the exchange.

11. (XXX/8.14)Suppose that arrays A and B are sorted and that each contains N elements. Give an O(log N) algorithm to find the median of $A \cup B$.

12. (21.8) IN the standard array representation of a binary tree that is used for binary heaps, a complete binary tree with N elements uses array positions 1 through N. More space is needed if the tree is not complete. What is the maximum array size needed to store an N-element binary tree (in this representation) whose height is two more than the height of the complete tree of N elements? What is the maximum array size needed to store any tree with N elements?

13. (7.9) Consider this method for computing Fibonacci numbers F_n = fib(n):

```
public static long fib (int n) {
   if (n <= 1)
      return n;
   else
      return fib(n-11)+fib(n-2);
```

Prove by induction that $F_1 + F_2 + \ldots + F_N = F_{N+2} - 1$

14. (7.12) C(N) represents the number of calls to fib during the calculation of F_N. A recurrence relation for C(N) is
$C(0) = C(1) = 1$, $C(N) = C(N-1) + C(N-2) + 1$ for $N \geq 2$.

Using this recurrence relation, you must show by (strong) induction that $C(N) = F_{N+2} + F_{N-1} - 1$ for all $N > 1$.

15. (15 points) Find a closed-form formula for $\sum_{i=1}^{k} i2^{i-1}$, and then prove the correctness of your formula by mathematical induction.

16. (5.20/5.26) Analyze the average cost (count the number of probes) of a successful search using binary search. . In order to make the calculations simpler, you may assume that $N = 2^k - 1$ for some integer $k \geq 0$, and that all elements are equally likely to be the one we are searching for. Note that the problem only deals with successful searches (where the item we are looking for is actually in the array).

Hint: For each *i*, count how many of the array elements will be found after binary search examines exactly *i* array elements. (For example, the middle element is the only one to be found after examining one array element, and there are two elements that can be found after two probes into the array. Use a summation to help you find the average.