CS361 Homework #4
Due Tuesday, November 21st

1. Consider an implementation of Union-Find where I use Union-by-Size: that is, I look at the size of the two components, and always have the representative of the larger one become the parent of the representative of the smaller one. Prove inductively that the depth of a component that contains ℓ items is at most $\log_2 \ell$, even without any path compression. (The depth of a component consisting of a single isolated item is zero.)

2. Consider a binary tree class defined as (in C++; in Java these pointers would be references instead)

```
class node {
  node *left; // NULL if no left daughter
  node *right; // NULL if no right daughter
  int key;
};
class tree {
  node *root; // NULL if empty
};
```

   Write C++ or Java code (or clear pseudocode) for a recursive function `bool isSearchTree(tree t)` that determines whether `t` is a valid binary search tree.

3. Suppose I have a binary tree. Define its *average depth* as the average depth of a leaf, where all the leaves are equally likely. (Let's say that the depth of a 1-node tree, consisting just of the root, is 0). For instance, the tree shown in Figure 1 has average depth 9/4.

   Prove by induction, by using the fact that a binary tree consists of two subtrees, that the average depth of any binary tree with n leaves is at least $\log_2 n$.

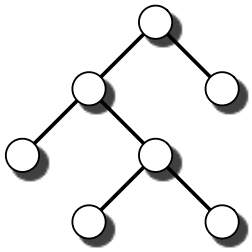

Figure 1: A tree with 4 leaves, whose depths are 2, 3, 3, and 1. Its average depth is 9/4.

4. Let x, y, and z be three Boolean variables. Write the constraint

$$x \text{ OR } y = z$$

   in SAT form, i.e., an AND ($\wedge$) of clauses, each of which is the OR ($\vee$) of two or three variables, which may be negated. Do the same thing for

$$x \text{ XOR } y = z \ .$$

5. Is this 2-SAT formula satisfiable?

$$(x \vee y) \wedge (\overline{y} \vee z) \wedge (\overline{x} \vee \overline{z}) \wedge (x \vee \overline{z})$$

   Either prove that it is not, or find a solution. In the latter case, state whether or not it is the only solution, and justify your answer.

6. Write the multiplication table mod 7. For each nonzero element, what is its inverse? In other words, for each x, what is the "$1/x$" such that $x \cdot 1/x = 1$?

7. What is $\log_3 4$ in the mod-7 world?

8. An element x is called a *generator* for the integers mod p if its powers include all the nonzero integers mod p. Which integers are generators for the integers mod 13?

9. Describe how to calculate x^{1000} with much fewer than 1000 multiplications.

10. Suppose you and I are doing Diffie-Helman key exchange. In order to exchange a 9-bit key, we publicly agree that we will operate mod 127, and that we will use 3 as a generator. I choose $a = 96$, and you choose $b = 40$. What numbers do we send back and forth, and what secret key do we end up with? Rather than just giving the answers, explain what computations we do, and try to do them with as few multiplications as possible.