CSE 311: Foundations of Computing I

Homework 6 (due November 14th at 11:59 PM)

Directions: *Write up carefully argued solutions to the following problems. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. You may use results from lecture, the theorems handout, and previous homeworks without proof.*

1. Parsley, Sage, Rosemary, and Running Time (20 points)

Let the function T, which describes the running time of a recursive algorithm, be defined as:

$$T(n) = 1 \qquad \text{if } 1 \leq n \leq 10$$
$$T(n) = T(\lfloor n/5 \rfloor) + 7 \cdot T(\lfloor n/10 \rfloor) + n \qquad \text{for all } n > 10$$

Prove by strong induction that, for all $n \geq 1$, we have $T(n) \leq 10n$.
Hint: The only fact about $\lfloor\ \rfloor$ that you will need is that when $x \geq 1$, $\lfloor x \rfloor$ is an integer, and $1 \leq \lfloor x \rfloor \leq x$.

2. Win Sum, Lose Sum (20 points)

Let S be the set defined as follows:

Basis Step: $5 \in S$; $9 \in S$;

Recursive Step: if $x, y \in S$, then $x + y \in S$.

Show that, for every integer $n \geq 32$, it holds that $n \in S$.
Hint: Strong (not structural) induction is the right tool here since the quantifier is over n (not S).

3. Hope Strings Eternal (20 points)

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

(a) [5 Points] Binary strings with length divisible by 3.

(b) [5 Points] Binary strings where every occurrence of a 0 is immediately followed by a 1.

(c) [5 Points] Binary strings that start with 1 and have even length.

(d) [5 Points] Binary strings with an odd number of 1s.

4. Say "Trees" (40 points)

Consider the following definition of a binary tree:

Bases Step: `Nil` is a **Tree**.

Recursive Step: If L is a **Tree** and R is a **Tree** and $x \in \mathbb{Z}$, then $\texttt{Tree}(x, L, R)$ is a **Tree**.

We define the set of values stored in a binary tree as follows:

$$\begin{array}{rcl} \mathsf{values}(\texttt{Nil}) & = & \{\} \\ \mathsf{values}(\texttt{Tree}(x, L, R))) & = & \mathsf{values}(L) \cup \{x\} \cup \mathsf{values}(R) \end{array}$$

We define the predicate BST on trees as follows:

$$\begin{array}{rcl} \mathsf{BST}(\texttt{Nil}) & = & \top \\ \mathsf{BST}(\texttt{Tree}(x, L, R))) & = & \mathsf{BST}(L) \wedge (\forall y.\ y \in \mathsf{values}(L) \rightarrow y < x) \wedge \\ & & \mathsf{BST}(R) \wedge (\forall z.\ z \in \mathsf{values}(R) \rightarrow x < z) \end{array}$$

The second part says that $\texttt{Tree}(x, L, R)$ is a binary search tree if every value stored in L is less than x, every value stored in R is greater than x, and L and R are themselves binary search trees.

Finally, for any $v \in \mathbb{Z}$, we define the function $\mathsf{contains}_v(T)$, which determines whether the BST T contains the value v, using the standard binary search tree algorithm, as follows:

$$\begin{array}{rcl} \mathsf{contains}_v(\texttt{Nil}) & = & \mathsf{F} \\ \mathsf{contains}_v(\texttt{Tree}(x, L, R))) & = & \begin{cases} \top & \text{if } x = v \\ \mathsf{contains}_v(L) & \text{if } v < x \\ \mathsf{contains}_v(R) & \text{if } x < v \end{cases} \end{array}$$

The more obvious way to define $\mathsf{contains}_v(T)$ would be just "$v \in \mathsf{values}(T)$". In this problem, your task is to prove that the above definition, using the binary search tree algorithm, is still correct.

Let $v \in \mathbb{Z}$. Use structural induction on T to prove that, if $\mathsf{BST}(T)$, then $\mathsf{contains}_v(T)$ iff $v \in \mathsf{values}(T)$. In addition, there are a few tricky points that are worth noting:

- You are proving an *implication* by induction. (You are not asked to prove anything about what the algorithm does if $\mathsf{BST}(T)$ is false.) Hence, the predicate that you define should be an implication.
- This is the most difficult proof we have given you to date. It would be a mistake to start it on the last day.
- (My solution to this problem is not long. In fact, it's shorter than this problem statement! Most of the solution is filling in the "easy parts" of the proof that follow from the structure of what you need to prove.)

5. Extra Credit: Magical Pebbles (0 points)

Consider an infinite sequence of positions $1, 2, 3, \ldots$ and suppose we have a pebble at position 1 and another pebble at position 2. In each step, we choose one of the pebbles and move it according to the following rule: Say we decide to move the pebble at position i; if the other pebble is not at any of the positions $i+1, i+2, \ldots, 2i$, then it goes to $2i$, otherwise it goes to $2i+1$.

For example, in the first step, if we move the pebble at position 1, it will go to 3 and if we move the pebble at position 2 it will go to 4. Note that, no matter how we move the pebbles, they will never be at the same position.

Use induction to prove that, for any given positive integer n, it is possible to move one of the pebbles to position n. For example, if $n = 7$ first we move the pebble at position 1 to 3. Then, we move the pebble at position 2 to 5 Finally, we move the pebble at position 3 to 7.