CSE 311: Foundations of Computing I

Homework 6 (due Wed, May 16, 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. Runtime... Better Go Catch It! (20 points)

Let $c > 0$ be an integer. The following recursive definition describes the running time of a recursive algorithm.

$$\begin{aligned} &T(0) = 0 \\ &T(n) \le c && \text{for all } n \le 20 \\ &T(n) = T\left(\left\lfloor \frac{3n}{4} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{5} \right\rfloor\right) + cn && \text{for all } n > 20 \end{aligned}$$

Prove by strong induction that $T(n) \le 20cn$ for all integers $n \ge 0$.
Hint: The only fact about $\lfloor \ \rfloor$ that you will need is that, when $x \ge 0$, $\lfloor x \rfloor$ is an integer and $0 \le \lfloor x \rfloor \le x$.

2. Happily Ever After (20 points)

Let S be the set defined as follows.

Basis Step: $4 \in S$; $7 \in S$

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

Show that, for all integers $n \ge 18$, we have $n \in S$.
Hint: Strong induction is the right tool here since the quantifier is not over S.

3. Putting the cat in reverse (20 points)

Consider the following recursive definition of strings.

Bases Step: "" is an string.

Recursive Step: If X is an string, then for any character c, append(X, c) is a string. (This is the string with c added to the end of X.)

Consider the following two functions defined on strings:

$$\begin{aligned} &\text{concat}(X, "") = X && \text{rev}("") = "" \\ &\text{concat}(X, \text{append}(Y, c)) = \text{append}(\text{concat}(X, Y), c) && \text{rev}(\text{append}(X, c)) = \text{concat}(\text{append}("", c), \text{rev}(X)) \end{aligned}$$

(a) [8 Points] Prove that concat is symmetric across "". That is, prove that for all strings X, we have

$$\text{concat}(X, "") = \text{concat}("", X)$$

(b) [12 Points] Prove that rev(concat(X, Y)) = concat(rev(Y), rev(X)) for all strings X and Y.

You may use, without proof, the fact that concat is associative: that is, for all strings X, Y, Z, we have

$$\text{concat}(\text{concat}(X, Y), Z) = \text{concat}(X, \text{concat}(Y, Z))$$

4. Proving BST Insertion Works! (20 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 is an integer, then $\texttt{Tree}(x, L, R)$ is a **Tree**.

The standard *BST insertion* function can be written as the following:

$$\begin{aligned}
\textbf{insert}(v, \texttt{Nil}) &= \texttt{Tree}(v, \texttt{Nil}, \texttt{Nil}) \\
\textbf{insert}(v, \texttt{Tree}(x, L, R))) &= \begin{cases} \texttt{Tree}(x, \textbf{insert}(v, L), R) & \text{if } v < x \\ \texttt{Tree}(x, L, \textbf{insert}(v, R)) & \text{otherwise.} \end{cases}
\end{aligned}$$

Next, define a program **less** which checks if an entire BST is less than a provided integer v:

$$\begin{aligned}
\textbf{less}(v, \texttt{Nil}) &= \texttt{true} \\
\textbf{less}(v, \texttt{Tree}(x, L, R)) &= x < v \;\; \texttt{and} \;\; \textbf{less}(v, L) \;\; \texttt{and} \;\; \textbf{less}(v, R)
\end{aligned}$$

Prove that, for all $b \in \mathbb{Z}$, $x \in \mathbb{Z}$ and all trees T, if $\textbf{less}(b, T)$ and $x < b$, then $\textbf{less}(b, \textbf{insert}(x, T))$. In English, this means that, given an upper bound on the elements in a BST, if you insert something that meets that upper bound, it is still an upper bound.

You should use structural induction on T for this question, but there are a few tricky bits that are worth pointing out up-front:

- You are proving an implication *by induction*. This means, in your Base Case, you assume the first part and prove the second one.
- Because of this, there will be two implications going on in your Induction Step. This can be very tricky. You will assume *both* your IH and the left side of what you're trying to prove. You will end up needing to use both of them at some point in your proof.
- This is the most difficult proof we have given you to date. It would be a mistake to start it on the last day.

5. Strings are Strings (20 points)

For each of the following, construct regular expressions that match the given set of strings:

(a) [5 Points] The set of all binary strings that end with 0 and have even length, or start with 1 and have odd length.

(b) [5 Points] The set of all binary strings with the number of 0s congruent to 2 modulo 3.

(c) [5 Points] The set of all binary strings that contain at least one 1 and at most two 0's.

(d) [5 Points] The set of all binary strings that don't contain 001.

You can submit and check your answers to this question using `https://grinch.cs.washington.edu/cse311/regex`.

6. 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.