CSE 311: Foundations of Computing I

Homework 6 (due November 15th at 11:00 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. However, you may use results from lecture, the theorems handout, and previous homeworks without proof.*

1. Leaps and Bounds (16 points)

Let $T(n)$ denote the running time of some algorithm, where n is the size of the input. Suppose that this running time satisfies the following equations:

$$T(n) = 8 \qquad \text{if } n = 1$$
$$T(n) = T(\lfloor n/2 \rfloor) + 9\lfloor n/3 \rfloor + 4 \qquad \text{for all } n > 1$$

Use strong induction to prove this *upper bound* on the running time: for all $n \geq 1$, we have $T(n) \leq 10n$.

Hint: The only facts about $\lfloor \cdot \rfloor$ that you will need are that, for all $x \in \mathbb{R}$, $\lfloor x \rfloor$ is an integer satisfying $\lfloor x \rfloor \leq x$ and, additionally, when $x \geq 1$, we also have $\lfloor x \rfloor \geq 1$.

(This running time could arise, for example, if the algorithm operates on an input of size n by, first, performing some work that takes $9\lfloor n/3 \rfloor + 4$ steps and then calling itself recursively on an input of half that size.)

2. And Then Sum (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 $n \geq 18$, we have $n \in S$.
Hint: Strong induction is the right tool here since the quantifier is not over S.

3. Some Strings Attached (20 points)

For each of the following, write a recursive definition of the set of strings satisfying the given properties. *Briefly* justify why your solution is correct.

(a) [5 Points] Binary strings $x \in \{0,1\}^*$ whose length $|x|$ satisfies $|x| \equiv 2 \pmod 3$.

(b) [5 Points] Binary strings such that none of the substrings of adjacent 1s (i.e. "$11\cdots1$") have odd length.[1]

(c) [5 Points] Binary strings that have at least one 1 and an even number of 0s.

(d) [5 Points] Strings of parentheses (i.e., over the alphabet $\Sigma = \{(,)\}^*$) where left and right parentheses occur in matched pairs that are properly nested. E.g, "(())()" is included but "())(()" is not: even though both have 3 left parentheses and 3 right parentheses, those in the second string are not properly nested.

[1] Here, I mean only substrings containing *all* the adjacent 1s. For example, "011110" is included because the substring of adjacent 1s has length 4. Technically, it also has "111" as a substring, but that substring does not contain all (four) adjacent 1s.

4. Tied Up With Strings [Online] (20 points)

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

(a) [4 Points] Binary strings $x \in \{0,1\}^*$ whose length $|x|$ satisfies $|x| \equiv 2 \pmod 3$.

(b) [4 Points] Binary strings that start with a 1 and have an even number of 0s.

(c) [4 Points] Binary strings that have at least one 1 and an even number of 0s.

(d) [4 Points] Binary strings with at least two 1s.

(e) [4 Points] Binary strings with at least two 1s **or** at most two 0s.

> Submit and check your answers to this question here:
>
> https://grinch.cs.washington.edu/cse311/regex
>
> Think carefully about your answer to make sure it is correct before submitting. You have only 5 chances to submit a correct answer.

5. Wish List (24 points)

We then define the set **Lists** as follows:

Basis Elements: null $\in$ **Lists**.

Recursive Step: for any $x \in \mathbb{Z}$, if $L \in$ **Lists**, then Node$(x, L) \in$ **Lists**.

These are lists (essentially, linked lists) with integer values stored in the nodes.

The following function, which takes two lists as arguments, appends the elements from the first list to the front of the second list but in reverse order:

$$\begin{array}{rcll} \text{shift}(\text{null}, R) & = & R & \text{for any } R \in \textbf{Lists} \\ \text{shift}(\text{Node}(x, L), R) & = & \text{shift}(L, \text{Node}(x, R)) & \text{for any } x \in \mathbb{Z} \text{ and } L, R \in \textbf{Lists} \end{array}$$

We can then define the following function that reverses a list:

$$\text{reverse}(L) = \text{shift}(L, \text{null}) \quad \text{for all } L \in \textbf{Lists}$$

Since the second list passed to shift is empty, the result is just the reversal of the first list.

Now, let $f : \mathbb{Z} \to \mathbb{Z}$ be a function that takes an integer input and returns an integer output. Then, we define the following function, which takes a list containing the numbers $x_1, x_2, \ldots$ and returns a list containing the numbers $f(x_1), f(x_2), \ldots$:

$$\begin{array}{rcll} \text{map}_f(\text{null}) & = & \text{null} & \\ \text{map}_f(\text{Node}(x, L)) & = & \text{Node}(f(x), \text{map}_f(L)) & \text{for any } x \in \mathbb{Z} \text{ and } L \in \textbf{Lists} \end{array}$$

(a) [4 Points] Let $f : \mathbb{Z} \to \mathbb{Z}$ be the function defined by $f(x) := 2x + 1$. Use the definitions to calculate reverse$(\text{map}_f(L))$, where L is the list Node$(1, \text{Node}(2, \text{Node}(3, \text{null})))$. Show your work.

(b) [16 Points] Prove that we have $\text{map}_f(\text{shift}(L, R)) = \text{shift}(\text{map}_f(L), \text{map}_f(R))$ for any $L, R \in$ **Lists** by structural induction on L.

(c) [4 Points] Prove that $\text{map}_f(\text{reverse}(L)) = \text{reverse}(\text{map}_f(L))$ for all $L \in$ **Lists**.

6. Extra Credit: Sticks And Stones (0 points)

Consider an infinite sequence of positions $1, 2, 3, \ldots$ and suppose we have a stone at position 1 and another stone at position 2. In each step, we choose one of the stones and move it according to the following rule: Say we decide to move the stone at position i; if the other stone 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 stone at position 1, it will go to 3 and if we move the stone at position 2 it will go to 4. Note that, no matter how we move the stones, 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 stones to position n. For example, if $n = 7$ first we move the stone at position 1 to 3. Then, we move the stone at position 2 to 5 Finally, we move the stone at position 3 to 7.