Section 08: Induction, Regular Expressions

1. Regular Expressions

(a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

(b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

(c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

(d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

(e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

2. CFGs

Write a context-free grammar to match each of these languages.

(a) All binary strings that end in 00.

(b) All binary strings that contain at least three 1's.

(c) All binary strings with an equal number of 1's and 0's.

(d) All binary strings of the form xy, where $|x| = |y|$, but $x \neq y$.

3. Structural Induction

(a) Consider the following recursive definition of strings.

Basis Step: "" is a string

Recursive Step: If X is a string and c is a character then $\mathsf{append}(c, X)$ is a string.

Recall the following recursive definition of the function len:

$$\begin{aligned} \mathsf{len}(\texttt{""}) &= 0 \\ \mathsf{len}(\mathsf{append}(c, X)) &= 1 + \mathsf{len}(X) \end{aligned}$$

Now, consider the following recursive definition:

$$\begin{aligned} \mathsf{double}(\texttt{""}) &= \texttt{""} \\ \mathsf{double}(\mathsf{append}(c, X)) &= \mathsf{append}(c, \mathsf{append}(c, \mathsf{double}(X))). \end{aligned}$$

Prove that for any string X, $\mathsf{len}(\mathsf{double}(X)) = 2\mathsf{len}(X)$.

(b) Consider the following definition of a (binary) **Tree**:

Basis Step: $\bullet$ is a **Tree**.

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

The function leaves returns the number of leaves of a **Tree**. It is defined as follows:

$$\begin{aligned} \mathsf{leaves}(\bullet) &= 1 \\ \mathsf{leaves}(\texttt{Tree}(\bullet, L, R)) &= \mathsf{leaves}(L) + \mathsf{leaves}(R) \end{aligned}$$

Also, recall the definition of size on trees:

$$\begin{aligned} \mathsf{size}(\bullet) &= 1 \\ \mathsf{size}(\texttt{Tree}(\bullet, L, R)) &= 1 + \mathsf{size}(L) + \mathsf{size}(R) \end{aligned}$$

Prove that $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$ for all Trees T.

(c) Prove the previous claim using strong induction. Define $P(n)$ as "all trees T of size n satisfy $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$". You may use the following facts:

- For any tree T we have $\mathsf{size}(T) \geq 1$.
- For any tree T, $\mathsf{size}(T) = 1$ if and only if $T = \bullet$.

If we wanted to prove these claims, we could do so by structural induction.

Note, in the inductive step you should start by letting T be an arbitrary tree of size $k+1$.

4. Walk the Dawgs

Suppose a dog walker takes care of $n \geq 12$ dogs. The dog walker is not a strong person, and will walk dogs in groups of 3 or 7 at a time (every dog gets walked exactly once). Prove the dog walker can always split the n dogs into groups of 3 or 7.

5. For All

For this problem, we'll see an incorrect use of induction. For this problem, we'll think of all of the following as binary trees:

- A single node.
- A root node, with a left child that is the root of a binary tree (and no right child)
- A root node, with a right child that is the root of a binary tree (and no left child)
- A root node, with both left and right children that are roots of binary trees.

Let $P(n)$ be "for all trees of height n, the tree has an odd number of nodes"

Take a moment to realize this claim is false.

Now let's see an incorrect proof:

We'll prove $P(n)$ for all $n \in \mathbb{N}$ by induction on n.

Base Case ($n = 0$): There is only one tree of height 0, a single node. It has one node, and $1 = 2 \cdot 0 + 1$, which is an odd number of nodes.

Inductive Hypothesis: Suppose $P(i)$ holds for $i = 0, ..., k$, for some arbitrary $k \geq 0$.

Inductive Step: Let T be an arbitrary tree of height k. All trees with nodes (and since $k \geq 0$, T has at least one node) have a leaf node. Add a left child and right child to a leaf (pick arbitrarily if there's more than one), This tree now has height $k+1$ (since T was height k and we added children below). By IH, T had an odd number of nodes, call it $2j+1$ for some integer j. Now we have added two more, so our new tree has $2j+1+2 = 2(j+1)+1$ nodes. Since j was an integer, so is $j+1$, and our new tree has an odd number of nodes, as required, so $P(k+1)$ holds.

By the principle of induction, $P(n)$ holds for all $n \in \mathbb{N}$. Since every tree has an (integer) height of 0 or more, every tree is included in some $P()$, so the claim holds for all trees.

(a) What is the bug in the proof?

(b) What should the starting point and target of the IS be (you can't write a full proof, as the claim is false).

6. Reversing a Binary Tree

Consider the following definition of a (binary) **Tree**.

Basis Step `Nil` is a **Tree**.

Recursive Step If L is a **Tree**, R is a **Tree**, and x is an integer, then $\texttt{Tree}(x, L, R)$ is a **Tree**.

The sum function returns the sum of all elements in a **Tree**.

$$\begin{aligned} \mathsf{sum}(\texttt{Nil}) &= 0 \\ \mathsf{sum}(\texttt{Tree}(x, L, R)) &= x + \mathsf{sum}(L) + \mathsf{sum}(R) \end{aligned}$$

The following recursively defined function produces the mirror image of a **Tree**.

$$\begin{aligned} \mathsf{reverse}(\texttt{Nil}) &= \texttt{Nil} \\ \mathsf{reverse}(\texttt{Tree}(x, L, R)) &= \texttt{Tree}(x, \mathsf{reverse}(R), \mathsf{reverse}(L)) \end{aligned}$$

Show that, for all **Tree**s T that

$$\mathsf{sum}(T) = \mathsf{sum}(\mathsf{reverse}(T))$$