0. Structural Induction: Strings

Recursive Definition of a String:
- Basis Step: $\epsilon$ is a string
- Recursive Step: If $w$ is a string and $a$ is a character, $w \cdot a$ is a string (the string $w$ with the character $a$ appended on to the end)

Recursive functions on String:
Length:

$$\begin{align*}
\text{len}(\epsilon) & = 0 \\
\text{len}(w \cdot a) & = \text{len}(a \cdot w) = 1 + \text{len}(w)
\end{align*}$$

Reverse:

$$\begin{align*}
\text{rev}(\epsilon) & = \epsilon \\
\text{rev}(w \cdot a) & = a \cdot \text{rev}(w)
\end{align*}$$

Prove that for any string $x$, $\text{len}(\text{rev}(x)) = \text{len}(x)$.

Complete the proof below:
Let $P(x)$ be "______________________________". We will prove $P(x)$ for all strings $x$ by structural induction on the set of strings.

Base Case: $(x = \epsilon)$

Let $s$ be an arbitrary string not covered by the base case. Then by the exclusion rule, $s = w \cdot a$ for some string $w$ and some character $a$.

Inductive Hypothesis: Suppose $P(______)$ holds.

Inductive Step: Goal: Show that $P(______)$ holds

This proves $P(______)$.

Conclusion: $P(x)$ holds for all strings $x$ by structural induction.
1. Structural Induction: CharTrees

Recursive Definition of CharTrees:

- **Basis Step:** Null is a **CharTree**
- **Recursive Step:** If $L, R$ are **CharTrees** and $c \in \Sigma$, then $\text{CharTree}(L, c, R)$ is also a **CharTree**

Intuitively, a **CharTree** is a tree where the non-null nodes store a char data element.

Recursive functions on CharTrees:

- The preorder function returns the preorder traversal of all elements in a **CharTree**.
  
  \[
  \text{preorder(Null)} = \varepsilon \\
  \text{preorder(CharTree}(L, c, R)) = c \cdot \text{preorder}(L) \cdot \text{preorder}(R)
  \]

- The postorder function returns the postorder traversal of all elements in a **CharTree**.
  
  \[
  \text{postorder(Null)} = \varepsilon \\
  \text{postorder(CharTree}(L, c, R)) = \text{postorder}(L) \cdot \text{postorder}(R) \cdot c
  \]

- The mirror function produces the mirror image of a **CharTree**.
  
  \[
  \text{mirror(Null)} = \text{Null} \\
  \text{mirror(CharTree}(L, c, R)) = \text{CharTree}(\text{mirror}(R), c, \text{mirror}(L))
  \]

- Finally, for all strings $x$, let the “reversal” of $x$ (in symbols $x^R$) produce the string in reverse order.

Additional Facts:
You may use the following facts:

- For any strings $x_1, \ldots, x_k$: $(x_1 \cdot \ldots \cdot x_k)^R = x_k^R \cdot \ldots \cdot x_1^R$
- For any character $c$, $c^R = c$

Statement to Prove:
Show that for every **CharTree** $T$, the reversal of the preorder traversal of $T$ is the same as the postorder traversal of the mirror of $T$. In notation, you should prove that for every **CharTree**, $T$: \([\text{preorder}(T)]^R = \text{postorder}(\text{mirror}(T))\).

There is an example and space to work on the next page.
Example for Intuition:

Let $T_i$ be the tree above.

$\text{preorder}(T_i) = \text{"abcd"}$. 

$T_i$ is built as $(\text{null}, a, U)$

Where $U$ is $(V, b, W)$,

$V = (\text{null}, c, \text{null})$, $W = (\text{null}, d, \text{null})$.

This tree is mirror($T_i$).

$\text{postorder}(\text{mirror}(T_i)) = \text{"dcba"}$,

"dcba" is the reversal of "abcd" so

$[\text{preorder}(T_i)]^R = \text{postorder(\text{mirror}(T_i))}$ holds for $T_i$. 

2. Structural Induction: Dictionaries

Recursive definition of a Dictionary (i.e. a Map):

- Basis Case: [] is the empty dictionary.
- Recursive Case: If D is a dictionary, and a and b are elements of the universe, then \((a \rightarrow b) :: D\) is a dictionary that maps a to b (in addition to the content of D).

Recursive functions on Dictionaries:

\[
\begin{align*}
\text{AllKeys}([],) &= [] \\
\text{AllKeys}((a \rightarrow b) :: D) &= a :: \text{AllKeys}(D) \\
\text{len}([],) &= 0 \\
\text{len}((a \rightarrow b) :: D) &= 1 + \text{len}(D)
\end{align*}
\]

Recursive functions on Sets:

\[
\begin{align*}
\text{len}([],) &= 0 \\
\text{len}(a :: C) &= 1 + \text{len}(C)
\end{align*}
\]

Statement to prove:
Prove that \(\text{len}(D) = \text{len}(\text{AllKeys}(D))\).
3. Structural Induction: CFGs

Consider the following CFG:

\[ S \rightarrow SS \mid 0S1 \mid 1S0 \mid \epsilon \]

Prove that every string generated by this CFG has an equal number of 1’s and 0’s.

**Hint 1:** Start by converting this CFG to a recursively defined set.
**Hint 2:** You may wish to define the functions \#_0(x), \#_1(x) on a string \( x \).