In this lab, we'll work on authoring proofs. Try to be as rigorous, yet concise in your proofs as possible. If you have any questions about levels of formality or formatting, please don't hesitate to ask!

## #1 Formal Proof

Formally prove the following claims over strings (refer to Sipser §0.2 for a definition of *string*). We will begin by showing that strings, under concatenation, form an algebraic structure known as a *monoid*.

**Definition 1** (Monoid)**.** *Let $M$ be a nonempty set with binary operation $(+) : M \times M \to M$. We call the tuple $(M, +)$ a monoid if:*

*(a) There exists a distinguished element $u \in M$, called that* identity *of $M$, such that for all $x \in M$ we have*

$$x + u = u + x = x$$

*(b) The binary operation is* associative*: for all $x, y, z \in M$, we have*

$$(x + y) + z = x + (y + z)$$

**Claim 1.** *Let $\Sigma$ be a nonempty alphabet. Prove that $\Sigma^\star$ (the set of all strings over $\Sigma$, as defined in Lab 1A) is a monoid under concatenation. Specifically, show that (i) the empty string $\epsilon$ is an identity, and (ii) concatenation is associative.*

(Hint: this should not require any trickery past unfolding definitions. You may refer to descriptions in Sipser §0.2 as "proof" of certain properties of strings.)

We next prove an important property of monoids that will be used in the claim that follows it.

**Claim 2** (Uniqueness of Identity)**.** *Let $u \in M$ be the identity of arbitrary monoid $(M, +)$. Then $u$ is* unique*: if another element $z \in M$ behaves as an identity (meeting the criteria in Definition 1.(a)), then $z$ must necessarily equal $u$.*

(Hint: Suppose another element, call it $z \in M$, is also an identity. Make an algebraic argument to show that $z$ and $u$ must be equal.)

Next, use the above claim to prove the following.

**Claim 3.** *Let $x$ and $y$ be strings. If $x$ is a prefix of $y$ and $y$ is a prefix of $x$ then $x = y$.*

(Hint: $x$ is a prefix of $y$ if a string $z$ exists where $xz = y$. Make this an algebraic argument based on this definition!)

## #2 Induction Revisited

Consider the following definitions regarding binary trees.

**Definition 2** (Binary Tree)**.** *A binary tree is either:*

- *A* leaf *or*
- *A* node *with two sub-trees, its children.*

*We define the* root *of a binary tree to be a node that is not the child of any other node in the tree.*

**Definition 3** (Level and Height)**.** *The* level *of a node in a binary tree is the length of the path from the root to that node. The* height *of a binary tree is the maximal level of any node in the tree. We also use "level" to denote the* set of all nodes *that share the same level in the tree.*

**Definition 4** (Complete and Perfect Trees)**.** *A tree is* complete *if each of its levels contain its maximal number of possible nodes. A tree is* perfect *if each node of the tree contains zero or two children.*

Now prove the following claim by structural induction.

**Claim 4.** *Let $h$ be the height of a complete, perfect binary tree. If $n \leq h$, then there are $2^n$ nodes at level $n$ of this tree.*

(Hint 1 (problem solving): I suggest you begin this problem by drawing pictures. What do complete, perfect binary trees look like for $h = 0$? What about $h = 1$ and $h = 2$? Extrapolate from there.)

(Hint 2 (proof strategy): the "length" of a path from the root to the root is 0, meaning the root is at level $n = 0$. *Do not start your induction at level 1.*)

# #3 Constructive Proof (Optional)

Chess is played on a $n \times n$ board (usually $n = 8$). Consider the rook piece, which can move in any number of squares in a cardinal (non-diagonal) direction. A *rook's tour* is a sequence of moves for a single rook that causes the piece to visit every square of the board exactly once. When considering a tour, we are free to place our piece on the board at any position initially. Also, we only consider a square visited if the piece ends its movement on that square.

Prove via induction that there exists a rook's tour for any chessboard of size $n \geq 1$.

(Hint 1: note that proving the existence of a rook's tour means constructing a rook's tour for an arbitrary board, i.e., designing an algorithm that generates the tour!)

(Hint 2: pay attention to the details! This is not as straightforward of a proof as you might think!)