Proof by Structural Induction (Rosen 5.3 p354) To prove a universal quantification over a recursively defined set:

Basis Step: Show the statement holds for elements specified in the basis step of the definition.

Recursive Step: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

Definition The set of natural numbers (aka nonnegative integers), $\mathbb{N}$, is defined (recursively) by:

Basis Step:	$0 \in \mathbb{N}$
Recursive Step:	If $n \in \mathbb{N}$ then $n+1 \in \mathbb{N}$ (where $n+1$ is integer addition)

The function $sumPow$ with domain $\mathbb{N}$, codomain $\mathbb{N}$, and which computes, for input i, the sum of the first i powers of 2 is defined recursively by $sumPow : \mathbb{N} \to \mathbb{N}$ with

Basis step: $sumPow(0) = 1$.

Recursive step: If $x \in \mathbb{N}$ then $sumPow(x+1) = sumPow(x) + 2^{x+1}$.

Fill in the blanks in the following proof of $\forall n \in \mathbb{N}\,(sumPow(n) = 2^{n+1} - 1)$:

Since $\mathbb{N}$ is recursively defined, we proceed by ____________________.

Basis case We need to show that ____________________. Evaluating each side: $LHS = sumPow(0) = 1$ by the basis case in the recursive definition of $sumPow$; $RHS = 2^{0+1} - 1 = 2^1 - 1 = 2 - 1 = 1$. Since $1 = 1$, the equality holds.

Recursive step Consider arbitrary natural number n and assume, as the ____________________ that $sumPow(n) = 2^{n+1} - 1$. We need to show that ____________________. Evaluating each side:

$$LHS = sumPow(n+1) \overset{\text{rec def}}{=} sumPow(n) + 2^{n+1} \overset{\text{IH}}{=} (2^{n+1} - 1) + 2^{n+1}.$$

$$RHS = 2^{(n+1)+1} - 1 \overset{\text{exponent rules}}{=} 2 \cdot 2^{n+1} - 1 = \left(2^{n+1} + 2^{n+1}\right) - 1 \overset{\text{regrouping}}{=} (2^{n+1} - 1) + 2^{n+1}$$

Thus, $LHS = RHS$. The structural induction is complete and we have proved the universal generalization.

Definition The set of linked lists of natural numbers L is defined:

Basis Step:	$[] \in L$
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$, then $(n, l) \in L$

Examples:

Definition The length of a linked list of natural numbers L, $len : L \to \mathbb{N}$ is defined by:

Basis Step:		$length([])$	$= 0$
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$, then	$length((n, l))$	

Examples:

Extra example: The function *prepend* : $L \times \mathbb{N} \to L$ that adds an element at the front of a linked list is defined:

Definition The function *append* : $L \times \mathbb{N} \to L$ that adds an element at the end of a linked list is defined:

Basis Step:	If $m \in \mathbb{N}$ then
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$ and $m \in \mathbb{N}$, then

Examples:

Claim: $\forall l \in L\ (\ length(\ append(l, 100)\ ) > length(l)\ )$

Proof: By structural induction on L, we have two cases:

Basis Step

1. **To Show** $length(\ append([], 100)\ ) > length([])$	Because [] is the only element defined in the basis step of L, we only need to prove that the property holds for [].
2. **To Show** $length(\ (100, [])\ ) > length([])$	By basis step in definition of *append*.
3. **To Show** $(1 + length([])) > length([])$	By recursive step in definition of *length*.
4. **To Show** $1 + 0 > 0$	By basis step in definition of *length*.
5. **To Show** T	By properties of integers
QED	Because we got to T only by rewriting **To Show** to equivalent statements, using well-defined proof techniques, and applying definitions.

Recursive Step

Consider an arbitrary: $l = (n, l')$, $l' \in L$, $n \in \mathbb{N}$, and we assume as the **induction hypothesis** that:

$$length(\ append(l', 100)\ ) > length(l')$$

Our goal is to show that $length(\ append((n, l'), 100)\ ) > length((n, l'))$ is also true. We evaluate each side of the candidate inequality:

$LHS = length(\ append((n, l'), 100)\ ) = length(\ (n, append(l', 100))\ )$ by the recursive definition of *append*

$= 1 + length(\ append(l', 100)\ )$ by the recursive definition of *length*

$> 1 + length(l')$ by the induction hypothesis

$= length((n, l'))$ by the recursive definition of *length*

$= RHS$