Week 3 Part B highlights

- Practice with properties of recursively defined sets and functions
- Prove and disprove properties of recursively defined sets and functions with structural induction
- Define linked lists: a recursively defined data structure
- Recursively define the set of natural numbers
- Relate proof by structural induction to mathematical induction

Tuesday

Recall: RNA is made up of strands of four different bases that match up in specific ways.
The bases are elements of the set $B = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$.

Definition The set of RNA strands S is defined (recursively) by:

Basis Step:	$\texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

Definition (Of a function, recursively) A function *rnalen* that computes the length of RNA strands in S is defined by:

		$rnalen : S$	$\to \mathbb{Z}^+$
Basis Step:	If $b \in B$ then	$rnalen(b)$	$= 1$
Recursive Step:	If $s \in S$ and $b \in B$, then	$rnalen(sb)$	$= 1 + rnalen(s)$

$rnalen(\texttt{ACU}) =$ ____

Definition: A function basecount that computes the number of a given base b appearing in a RNA strand s is defined recursively: fill in codomain and sample function applications

$$basecount : S \times B \to$$

Basis Step: *If* $b_1 \in B, b_2 \in B$ $\quad basecount(b_1, b_2) = \begin{cases} 1 & \textit{when } b_1 = b_2 \\ 0 & \textit{when } b_1 \neq b_2 \end{cases}$

Recursive Step: *If* $s \in S, b_1 \in B, b_2 \in B$ $\quad basecount(sb_1, b_2) = \begin{cases} 1 + basecount(s, b_2) & \textit{when } b_1 = b_2 \\ basecount(s, b_2) & \textit{when } b_1 \neq b_2 \end{cases}$

$basecount(ACU, A) =$ ____

$basecount(ACU, G) =$ ____

Prove or disprove $\forall s \in S\,(rnalen(s) \geq basecount(s, \texttt{A}))$*:*

Note: Universal generalization is not enough to prove this claim!

Prove or disprove $\forall s \in S\,(rnalen(s) \geq basecount(s, \texttt{A}))$*:*

Proof*: By structural induction on _____, choose arbitrary* $s \in S$

Base case*: Assume* $s = \texttt{A} \vee s = \texttt{C} \vee s = \texttt{U} \vee s = \texttt{G}$.

Need to show $rnalen(s) \geq basecount(s, \texttt{A})$.

Case 1: To show $(s = \texttt{A}) \rightarrow (\ rnalen(s) \geq basecount(s, \texttt{A})\ )$.

Case 2: To show $(s = \texttt{C} \vee S = \texttt{U} \vee S = \texttt{G}) \rightarrow (\ rnalen(s) \geq basecount(s, \texttt{A})\ )$.

Proof by universal generalization*: To prove that* $\forall x\, P(x)$ *is true, we can take an arbitrary element* e *from the domain and show that* $P(e)$ *is true, without making any assumptions about* e *other than that it comes from the domain.*
New! Proof by Structural Induction *(Rosen 5.3 p354) To prove a universal quantification over a recursively defined set:*

> **Base Case***: Show the statement holds for elements specified in the basis step of the definition.*
>
> **Inductive Case***: 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.*

Inductive case: *Assume* $s = eb$, *for some* $e \in S$ *and* $b \in B$

Assume, as the **induction hypothesis** *that*

$$\textit{rnalen(e)} \geq \textit{basecount}(e, \texttt{A})$$

Need to show

$$\textit{rnalen(eb)} \geq \textit{basecount}(eb, \texttt{A})$$

Case 1: Want to show $(b = \texttt{A}) \rightarrow ($ *rnalen(eb)* $\geq$ *basecount*$(eb, \texttt{A})$ $)$.

Case 2: Want to show $(b = \texttt{C} \vee b = \texttt{U} \vee b = \texttt{G}) \rightarrow ($ *rnalen(eb)* $\geq$ *basecount*$(eb, \texttt{A})$ $)$.

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 $l \in L$, we have two cases:*

Base Case: $l = []$

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.*

Inductive Case: $l = (n, l')$, $l' \in L$, $n \in \mathbb{N}$, *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*

$= RHS$

Thursday

Invariant: *A property that is true about our algorithm no matter what.* *Rosen p375*

Theorem: *Statement that can be shown to be true, usually an important one.* *Rosen p81*

Less important theorems can be called **proposition**, **fact**, **result**.

A less important theorem that is useful in proving a theorem is called a **lemma**.

A theorem that can be proved directly after another one has been proved is called a **corollary**

Theorem: *A robot on an infinite 2-dimensional integer grid starts at* $(0,0)$ *and at each step moves to diagonally adjacent grid point. This robot can / cannot (circle one) reach* $(1,0)$.

Definition *The set of positions the robot can visit* P *is defined by:*

Basis Step: $(0,0) \in P$
Recursive Step: *If* $(x,y) \in P$, *then* *are also in* P

Lemma: $\forall (x,y) \in P, (x+y \text{ is an even integer})$

Proof of theorem using lemma: *To show is* $(1,0) \notin P$.

Rewriting the lemma to explicitly restrict the domain of the universal, we have
$\forall (x,y)\ (\ (x,y) \in P \to (x+y \text{ is an even integer})\ )$

Rewriting the lemman once more to its logically equivalent contrapositive form, we get
$\forall (x,y)\ (\ (x+y \text{ is an odd integer}) \to (x,y) \notin P\ )$ $\ldots(a)$

By universal instantiation on statement (a) *for domain element* $(1,0)$ *we get*
$(1+0 \text{ is an odd integer}) \to (1,0) \notin P$ $\ldots(b)$

Further, $1+0$ *evaluates to 1, which is an odd integer* $\ldots(c)$

Therefore, we can conclude that $(1,0) \notin P$ *by Modus Ponens on statements* (b) *and* (c) ■

Proof of lemma by structural induction:

Basis Step

Recursive Step. *Consider arbitrary* $(x, y) \in P$. *To show is:*

$$(x + y \text{ is an even integer}) \rightarrow (\text{sum of coordinates of next position is even integer})$$

Assume **as the induction hypothesis, IH** *that:*

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)*

Recall that the set of linked lists of natural numbers L *Basis Step:* $[] \in L$ *Recursive Step: If* $l \in L$ *and* $n \in \mathbb{N}$ *then* $(n, l) \in L$	*Recall that length of a linked list of natural numbers L,* $length : L \to \mathbb{N}$ *is defined by:* *Basis step:* $length([]) = 0$ *Recursive step: If* $l \in L$ *and* $n \in \mathbb{N}$ *then* $length((n, l)) = 1 + length(l)$

Prove or disprove: $\forall n \in \mathbb{N}\ \exists l \in L\ (\ length(l) = n\ )$

"New"! Proof by Mathematical Induction *(Rosen 5.1 p329)*
To prove a universal quantification over the set of all integers greater than or equals some base integer b:

Basis Step: *Show the statement holds for b.*

Recursive Step: *Consider an arbitrary integer n greater than or equal to b, assume (as the* **induction hypothesis**) *that the property holds for n, and use this and other facts to prove that the property holds for n + 1.*

The function $sumPow$ **with domain** $\mathbb{N}$**, codomain** $\mathbb{N}$ **computes, for input** i**, the sum of the first** i **powers of** 2, *and is recursively defined as*
$sumPow : \mathbb{N} \rightarrow \mathbb{N}$ *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 the domain is the set of natural numbers, 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:

Extra example Connect the function $sumPow$ *to binary expansions of positive integers.*

Definition *The exponent function on* $\mathbb{Z}^+$ *is defined by:*

$$\begin{array}{llll} \textit{Basis Step:} & \textit{If } n = 1 \textit{ then} & 2^n & = 2 \\ \textit{Recursive Step:} & \textit{If } n \in \mathbb{Z}^+, \textit{ then} & 2^{n+1} & = 2 \cdot 2^n \end{array}$$

Definition *The factorial function on* $\mathbb{Z}^+$ *is defined by:*

$$\begin{array}{llll} \textit{Basis Step:} & \textit{If } n = 1 \textit{ then} & n! & = 1 \\ \textit{Recursive Step:} & \textit{If } n \in \mathbb{Z}^+, \textit{ then} & (n+1)! & = (n+1) \cdot n! \end{array}$$

Prove or disprove: $\forall n \in \mathbb{Z}^+ \; (2^n < n!)$

$\mathbb{N}$	*The set of natural numbers*	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}^{\geq b}$	*The set of integers greater than or equal a basis element* b	$\{b, b+1, b+2, b+3, \ldots\}$

Prove or disprove: $\forall n \in \mathbb{Z}^{\geq 4}\ (2^n < n!)$

Proof: *By mathematical induction on* $n \in \mathbb{Z}^{\geq 4}$

Basis step*:*

Inductive step: *Consider* *and assume as* **inductive hypothesis** *that*

Want to show (WTS)