Name: Wen Chuan Lee
Student ID: 4927941 (leex7095)
Class: CSCI 2041
Title: Homework 5 - Solutions

Question 1: Power function, over natural numbers

We would like to show that:

$$\forall n : \; power\ n\ x \; = \; x^n$$

The Principle of Induction for natural numbers is:

$$\forall n,\ P(n)\ if\ P(0)\ and\ P(n-1) \Rightarrow\ P(n)$$

Our property P is:

$$P(n,x)\ is\ power\ n\ x \; = \; x^n$$

Our inductive proof would have 2 cases:

Base case: $P(0,x)$: show that $power\ (0,x) \; = \; 1$

$power\ 0\ x \; = \; 1.0$ (by definition of power)

Step: $P(n,x)$: show that $power\ n\ x \; = \; x^n$

Our Inductive Hypothesis is where $power\ (n\ - 1)\ x \; = x^{(n-1)} *. \ x^{(n-2)} \ *. \ x^{(n-3)} \ *. \ ..... \ *. \ x^0$ holds.

$power\ n\ x \; = \; x \; *. \ power\ (n-1)\ x$ (by definition of *power*)

$= \; x \; *. \ x^{(n-1)} *. \ x^{(n-2)} \ *. \ x^{(n-3)} \ *. \ ..... \ *. \ x^0$ (by inductive hypothesis)

$= \; x^n$ (by properties of multiplication)

Hence we have proven $\forall n : \; power\ n\ x \; = \; x^n$

Question 2: Power over structured numbers

We would like to show that:

$$\forall n \in nat,\ power\ n\ x\ =\ x^{\,to\text{–}Int\ (n)}$$

The Principle of Induction for type *nat* is:

$$\forall n,\ P(n)\ if\ P(Zero)\ and\ P(Succ\ n)\ when\ \ P(n)\ holds.$$

Our property P (for type *nat*) is:

$$P(n,x)\ is\ power\ n\ x\ =\ x^{\,to\text{–}Int\ (n)}$$

Our inductive proof would have 2 cases:

P(Zero,x): show that $power\ zero\ x\ \ =\ x^{to\text{–}Int\ (zero)}$

$$\begin{aligned} power\ Zero\ x\ &=\ 1.0 && \text{(by definition of power)} \\ &=\ x^0 && \text{(by mathematical definition of power)} \\ &=\ x^{\,to\text{–}Int\ (Zero)} && \text{(by definition of to\_Int)} \end{aligned}$$

P(n, x) : show that $power\ (Succ\ n)\ x\ =\ x^{\,to\text{–}Int\ (Succ\ n)}$

Our inductive hypothesis is $power\ n\ x\ =\ x^{\,to\text{–}Int\ (n)}$

$$\begin{aligned} power\ (Succ\ n)\ x\ &=\ x\ *.\ power\ n\ \ x && \text{(by definition of } \textit{power}\text{)} \\ &=\ x\ *.\ x^{\,to\text{–}Int\ n} && \text{(by inductive hypothesis)} \\ &=\ x^{\,to\text{–}Int\ n\ +\ 1} && \text{(by mathematical power)} \\ &=\ x^{\,to\text{–}Int\ (Succ\ n)} && \text{(by definition of to\_Int)} \end{aligned}$$

Hence we have proven $power\ n\ x\ =\ x^{\,to\text{–}Int\ (n)}$

Question 3: Length of Lists

The principle of induction for list type is:

$$\forall l,\ P(l)\ if\ P([\,])\ and\ P(v\ ::\ l'),\ if\ P(l')$$

We need to show the following property P:

$$P\ (l,r)\ =\ length\ (l\ @\ r)\ =\ length\ l\ +\ length\ r$$

Our base case is: $P\ (\,[\,],\ r)\ :\ show\ that\ \ length\ ([\,]\ @\ r\,)\ =\ length\ [\,]\ +\ length\ r$
from definition of function, we know that $P[\,]\ \Rightarrow\ 0$

$length\ (\,[\,]\ @\ r\,)\ =\ length\ r$ (by the function definition and properties of list: $[\,]\ @\ l\ =\ l$)

$=\ 0\ +\ length\ r$ (by the properties of addition)

$=\ length\ [\,]\ +\ length\ r$ (by definition of length)

Step: $P\ (x :: xs\ ,\ r)\ :\ show\ that\ length\ (x : xs\ @\ r)\ =\ length\ (x :: xs)\ +\ length\ r$

Our inductive hypothesis is $length\ (\ xs\ @\ r\ )\ =\ length\ xs\ +\ length\ r$

$length\ (x : xs\ @\ r)\ =\ length\ (x :: xs\ @\ r)$ (by properties of lists)

$=\ 1\ +\ length\ (xs\ @\ r)$ (by definition of sum)

$=\ 1\ +\ length\ xs\ +\ length\ r$ (by inductive hypothesis)

$=\ length\ (x : xs\ )\ +\ length\ r$ (by definition of sum)

Hence we have proven that $length\ (l\ @\ r)\ =\ length\ l\ +\ length\ r$.

Question 4: List length and reverse

The principle of induction for list type is:

$$\forall l,\ P(l)\ if\ P([\,])\ and\ P(v\ ::\ l'),\ if\ P(l')$$

We need to show the following property P:

$$P\ (l, r)\ =\ length\ (reverse\ l@r)\ =\ length\ l : r$$

Base: $P\ (\ [\,],\ r)\ :\ show\ that\ \ length\ (reverse\ [\,]\ @\ r\ )\ =\ length\ ([\,]@r)$

$length\ (reverse\ [\,]\ @\ r)$	$= length\ (r\ @\ [\,])$	(by definition of reverse)
	$=\ length\ r$	(by list properties)
	$=\ 0\ +\ length\ r$	(by mathematical operation)
	$=\ length\ [\,]\ +\ length\ r$	(by proof from Question 3)
	$=\ length\ ([\,]\ @\ r)$	(by definition of length)

Step: $P(x :: xs)\ :\ show\ that\ length\ (reverse\ x :: xs)\ = length\ (x : xs)$

The inductive hypothesis is $length\ (reverse\ xs)\ =\ length\ xs$

$length\ (reverse\ x :: xs)$	$=\ length\ (reverse\ xs\ @\ [x])$	(by definition of reverse)
	$=\ length\ (reverse\ xs\ )\ +\ length\ ([x])$	(by definition of length)
	$=\ length\ (reverse\ xs)\ +\ 1$	(by definition of length)
	$= 1 +\ length\ (xs)$	(by inductive hypothesis)
	$=\ length\ (x : xs)$	(by definition of length)

Therefore, we have proven that $length\ (reverse\ l@r)\ =\ length\ l : r$

Question 5: List length and reverse

The principle of induction for list type is:

$$\forall l,\ P(l)\ if\ P([\ ])\ and\ P(v\ ::\ l'),\ if\ P(l')$$

We need to show the following property P:

$$P\ (l1, l2)\ =\ \ reverse\ (l1@l2)\ =\ reverse\ l2\ @\ reverse\ l1$$

Base: $P\ (\ [\ ],\ l2)\ :\ show\ that\ \ reverse\ ([\ ]\ @\ l2\ )\ =\ reverse\ l2\ @\ reverse\ [\ ]$

$reverse\ (\ [\ ]\ @\ l2\ )$	$=\ reverse\ l2$	(by properties of lists)
	$=\ reverse\ l2\ @\ [\ ]$	(by properties of lists)
	$=\ reverse\ l2\ @\ reverse\ [\ ]$	(by definition of reverse)

Step: $P(x : xs,\ r)\ :\ show\ that\ reverse\ (x : xs\ @\ r\ )\ =\ reverse\ r\ @\ reverse\ (x : xs)$

By Induction Hypothesis we have $reverse\ (xs\ @\ r)\ =\ reverse\ r\ @\ reverse\ xs$

$reverse\ (x : xs\ @\ r)$	$=\ reverse\ (x :: xs\ @\ r\ )$	(by properties of lists)
	$=\ reverse\ (x\ ::\ (xs\ @\ r)\ )$	(by properties of lists)
	$=\ reverse\ \ (\ xs\ @\ \ r\ )\ @\ [x]$	(by definition of reverse)
	$=\ reverse\ r\ @\ reverse\ xs\ @\ [x]$	(by inductive hypothesis)
	$=\ reverse\ r\ @\ reverse\ (xs\ @\ [x])$	(by properties of lists)
	$=\ reverse\ r\ @\ reverse\ (x\ ::\ xs)$	(by definition of reverse)
	$=\ reverse\ r\ @\ reverse\ (x\ :\ xs)$	

Thus we have proven that $reverse\ (l1\ @\ l2)\ =\ reverse\ l2\ @\ reverse\ l1$

Question 6: Sorted Lists

We need to show that: $sorted\ l \Rightarrow sorted\ (place\ e\ l\ )$ (implication is true)

We need to show the following property P:

$$P\ (l1, e)\ =\ sorted\ l1 \Rightarrow sorted\ (place\ e\ l)$$

Base: $P\ (\ [\ ],\ e)\ :\ show\ that\ sorted\ [\ ]\ =\ sorted\ (place\ e\ [\ ])$

$sorted\ [\ ]\ =\ true$ (by definition of sorted)

$sorted\ (place\ e\ [\ ])\ =\ sorted\ (\ [\ e\ ]\ )$ (by definition of place)

$=\ sorted\ (\ e\ ::\ [\ ]\ )$ (by list properties)

$=\ true$ (by definition of sorted)

Hence we know that $sorted\ [\ ]\ \Rightarrow\ sorted\ (\ place\ e\ [\ ]\ )\ holds\ for\ the\ base\ case\ (empty\ list)$.

Step: $P(x :: xs,\ e)\ :\ show\ that\ sorted\ (\ x\ :: xs\ )\ \Rightarrow\ sorted\ (place\ e\ x :: xs)$ holds.

By the inductive hypothesis we are given that $sorted\ xs\ \Rightarrow\ sorted\ (place\ e\ xs)$ holds.

We break the inductive case down into a few sub-cases:

Case: x is only item in the list and e >= x

$sorted\ x\ =\ sorted\ (x ::\ [\ ]\ )\ =\ true$ (by definition of sorted)

$sorted\ (place\ e\ x\ ::\ [\ ])\ =\ sorted\ (x\ ::\ place\ e\ [\ ]\ )$ (by definition of place)

$=\ sorted\ (x\ @\ e\ ::\ [\ ]\ )$ (by definition of place and list properties)

$=\ true$ (since x <= e and definition of sorted)

Hence as both conditionals are true we have *true* by logical implication.

(*proof continued on next page)*

Case: x is only item in the list and e < x

$sorted\, x \;=\; sorted\,(x :: \;[\;]\;) \;=\; true$ (by definition of sorted)

$sorted\,(place\; e\; x \;::\; [\;]) \;=\; sorted\,(e \;::\; x \;::[\;])$ (by definition of place)

$\quad = \; sorted\,(x \;::\; e\;)$ (by definition of place and list properties)

$\quad = \; true$ (since x < e and definition of sorted)

Hence as both conditionals are true we have *true* by logical implication.

Case: x is NOT the only item in the list, list sorted, e >= x

$sorted\, x \;=\; sorted\,(x :: \; xs\;) \;=\; true$ (by definition of sorted and our case)

$sorted\,(place\; e\;\; x :: xs) \;=\; sorted\,(\,x :: place\; e\; xs)$ (by definition of place and our case)

$\quad = \; true$ (by inductive hypothesis as sorted x::xs is true)

Hence as both conditionals are true we have *true* by logical implication.

Case: x is NOT the only item in the list , list sorted, e < x

$sorted\, x \;=\; sorted\,(x :: \; xs\;) \;=\; true$ (by definition of sorted and our case)

$sorted\,(place\; e\;\; x :: xs) \;=\; sorted\,(\,e \;::\; x \;::\; xs)$ (by definition of place and our case)

$\quad = \; (e \;<=\; x \;\;\&\&\; sorted\,(x \;\; :: xs))$ (by definition of sorted and our case)

$\quad = \; true\;\&\; true$ (as sorted (x::xs) is true and e < x)

$\quad = \; true$ (by truth logic)

Hence as both conditionals are true we have *true* by logical implication.

Case: x is NOT the only item in the list, list NOT sorted, e >= x

$sorted\, x \;=\; sorted\,(x :: \; xs\;) \;=\; false$ (by definition of sorted and our case)

$sorted\,(place\; e\;\; x :: xs) \;=\; sorted\,(\,x :: place\; e\; xs)$ (by definition of place and our case)

$\quad = \; false$ (by inductive hypothesis as sorted x::xs is true)

Hence as both conditionals are false we have *true* by logical implication.

(proof continued on next page)

Case: x is NOT the only item in the list , list NOT sorted, e < x

$sorted\, x \;=\; sorted\,(x :: \; xs\,) \;=\; false$ (by definition of sorted and our case)

$sorted\,(place\; e\;\; x :: xs) \;\; = \; sorted\,(\; e \;\; :: \;\; x \;\; :: \;\; xs)$ (by definition of place and our case)

$= \; (e \;\; <= \;\; x \;\; \&\&\; sorted\,(x \;\; :: xs))$ (by definition of sorted and our case)

$= \; true \;\&\&\; false$ (as sorted (x::xs) is false and e < x)

$= \; false$ (by truth logic)

Hence as both conditionals are true we have *true* by logical implication.

$Therefore,\; based\; on\; all\; the\; sub-cases\; that\; handle\; all\; cases\; exhaustively,\; we\; have\; extensively\; proved\; that:$

$sorted\; l \;\Rightarrow\; sorted\,(place\; e\; l)$ will hold.

Question 7: Sorted Lists

The proof $is\_elem\; e\;(place\; e\; l)$ can be claimed without requiring that the list be sorted despite the fact that is_elem assumes the list is sorted (by definition of the function is_elem) because of the definition of the function place. Aside from the formal proof supplied in the sample proofs/ class lectures that the claim is true, we can safely claim that an element *e* exists in a list where *e* is already placed in the list because the function place will either place *e* in front of the list is e < x or else pass it down the list recursively. Despite the possibility of having the list *l* not be properly sorted, it e would have been compared to other elements in the list and put in the right position of the list where <u>e is placed in a position where it is either less than all the items after it, or more than and equal all the items after it, regardless of whether or not list l is actually sorted.</u>

As such, when is_elem checks is the element *e* exists within the list *l*, it will never fall into the case where there is an empty list that e is not in (as the function place never evaluates to an empty list, and even at it's most basic case, only element e will exist within the list). Also, the function is_elem will then go through every element in the list to check if any of the elements at each position have a value that is equal to element *e*. If e does not equal to the first item in the

list, it will recursively continue down the list until it finds the element e, which will happen as the requirement that e > x will always hold as the function place has already properly placed the element e into the list in the correct position. Hence the claim that:
$is\_elem\ e\ (place\ e\ l)$ will always hold true.

Question: Is the premise **sorted l** needed in the proof for question 6? Explain.

The premise that **sorted l** is also **not** needed in the proof for question 6 as the implication will hold whether or not that sorted l evaluates to true $sorted\ l \Rightarrow sorted\ (place\ e\ l)$. This is because that for all cases that sorted l evaluates to true, we have proven that $sorted\ (place\ e\ l)$ will also evaluate to true and also <u>never</u> evaluate to false, which finally $sorted\ l \Rightarrow sorted\ (place\ e\ l)$ evaluates to a true in the entire statement by logical implication. Even in the cases where $sorted\ l$ evaluates to false, and $sorted\ (place\ e\ l)$ evaluates to false, the implication that $sorted\ l \Rightarrow sorted\ (place\ e\ l)$ still evaluates to true by logical implication.

To further this argument, logical implications only evaluate to false only in the case where the first statement is true and the second statement is false, which will never occur in the implication that $sorted\ l \Rightarrow sorted\ (place\ e\ l)$, as we have shown in the proof in question 6.

Question 8: Correctness of imperative programs
