CIS 500 — Software Foundations

Midterm I

February 16, 2011

Name: ____________________________________________________________

Pennkey: _________________________________________________________

Scores:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Total (78 max)</td>
<td></td>
</tr>
</tbody>
</table>
1. (10 points) Consider the following Inductive definition:

\[
\text{Inductive ptree (X:Type) : Type :=}
\]
\[
| \ c1 : X \to X \to ptree X \\
| \ c2 : ptree X \to ptree X \to ptree X.
\]

Implicit Arguments c1 [[X]].
Implicit Arguments c2 [[X]].

For each of the following types, define a function (using Definition or Fixpoint) with the given type.

(a) \(\text{nat} \to \text{nat} \to \text{ptree} \text{nat}\)

(b) \(\forall X \ Y : \text{Type}, \ ptree X \to (X \to Y) \to ptree Y\)
2. (8 points) Recall the definition of \( \lor \) from Logic.v:

\[
\text{Inductive or (P Q : Prop) : Prop := } \\
| \lor\text{-intr} : P \rightarrow or P Q \\
| \lor\text{-intr}r : Q \rightarrow or P Q.
\]

Notation "P \lor Q" := (or P Q) : type_scope.

Write down a term of type \( \forall (P Q R:\text{Prop}), (P \lor Q \rightarrow R) \rightarrow Q \rightarrow R \).
3. (8 points) Recall the inductively defined proposition \( \text{le} \) from \texttt{Logic.v}:

\[
\begin{align*}
\text{Inductive} \ & \text{le} \ (n:\text{nat}) : \text{nat} \rightarrow \text{Prop} := \\
| \ & \text{le}_n : \text{le} \ n \ n \\
| \ & \text{le}_S : \text{forall} \ m, (\text{le} \ n \ m) \rightarrow (\text{le} \ n \ (S \ m)).
\end{align*}
\]

(a) What is the type of the \( \text{le}_n \) constructor? (I.e., what is printed if we send Coq the command \texttt{Check le\_n}?)

(b) Write down a term whose type is

\[
\text{forall} \ (n:\text{nat}), \text{le} \ 2 \ n \rightarrow \text{le} \ 2 \ (S \ (S \ n)).
\]
4. (14 points) Recall that a list \( l_3 \) is an “in-order merge” of lists \( l_1 \) and \( l_2 \) if it contains all the elements of \( l_1 \), in the same order as \( l_1 \), and all the elements of \( l_2 \), in the same order as \( l_2 \), with elements from \( l_1 \) and \( l_2 \) interleaved in any order. For example, the following lists (among others) are in-order merges of \([1,2,3]\) and \([4,5]\):

\[
\begin{align*}
[1,2,3,4,5] \\
[4,5,1,2,3] \\
[1,4,2,5,3]
\end{align*}
\]

Complete the following inductively defined relation in such a way that \texttt{merge} \( l_1 \ l_2 \ l_3 \) is provable exactly when \( l_3 \) is an in-order merge of \( l_1 \) and \( l_2 \).

\[
\text{Inductive merge \{X:Type\} : list X -> list X -> list X -> Prop :=}
\]
5. (16 points) A list $l_1$ is a permutation of another list $l_2$ if $l_1$ and $l_2$ have exactly the same elements (with each element occurring exactly the same number of times), possibly in different orders. For example, the following lists (among others) are permutations of the list $[1,1,2,3]$:

$$
[1,1,2,3] \\
[2,1,3,1] \\
[3,2,1,1] \\
[1,3,2,1]
$$

On the other hand, $[1,2,3]$ is not a permutation of $[1,1,2,3]$, since 1 does not occur twice.

Complete the following inductively defined relation in such a way that $\text{permutation } l_1 l_2$ is provable exactly when $l_1$ is a permutation of $l_2$. Feel free to create other inductive definitions besides $\text{permutation}$ if you find it helpful.

$$\text{Inductive permutation } \{X:\text{Type}\} : \text{list } X \to \text{list } X \to \text{Prop} := \text{ }$$
6. (8 points) Here is an induction principle for an inductively defined type myT.

\[
\text{myT\_ind} : \\
\text{forall } (X : \text{Type}) (P : \text{myT} \rightarrow \text{Prop}), \\
\text{(forall } x : X, P (c1 x)) \rightarrow \\
\text{(forall } s : \text{myT}, P s \rightarrow \text{forall } t : \text{myT}, P t \rightarrow P (c2 s t)) \\
\text{forall } t : \text{myT}, P t
\]

What is the definition of myT?
7. (16 points) Recall the definition of \texttt{double}:

\begin{verbatim}
Fixpoint double (n:nat) :=
  match n with
  | 0  => 0
  | S n' => S (S (double n'))
  end.
\end{verbatim}

Write an informal proof of this theorem:

\textit{Theorem}: For any natural numbers \(n\) and \(m\), if \(\texttt{double}\ n = \texttt{double}\ m\), then \(n = m\).