CIS 500 — Software Foundations

Midterm I

February 15, 2012

Name: 

Pennkey: 

<table>
<thead>
<tr>
<th>Scores</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>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Total (80 max)</td>
<td></td>
</tr>
</tbody>
</table>
1. (8 points) A 2-3 tree is a tree data structure in which (1) every node is labeled with a value (drawn from some set X), and (2) every node has zero, two, or three children. For example, here is a 2-3 tree of numbers:

![2-3 Tree Diagram]

(a) Complete the following inductive definition of 2-3 trees:

\[
\text{Inductive ttree \{X : Type\} : Type :=}
\]

(b) Write down a term of type \texttt{ttree nat} representing the tree shown above.
2. (6 points) Briefly explain the behavior of the \texttt{apply} and \texttt{apply... with...} tactics in Coq.

3. (6 points) For each of the following types, define a function (using \texttt{Definition} or \texttt{Fixpoint}) with the given type.

(a) \texttt{nat \rightarrow list (list nat)}

(b) \texttt{forall X Y : Type, list X \rightarrow (X \rightarrow Y) \rightarrow list Y}
4. (8 points) Write down the type of each of the following expressions. (For example, for the expression

    fun (x y : nat) => beq_nat (x+y) 10

you’d write nat -> nat -> bool.) If an expression is not typeable, write “ill typed.”

(a) fun (x : nat) => x :: []

(b) (2 :: 3 :: []) :: []

(c) fun (X : Type) (l : list X) =>
    match l with
    [] => []
    | h :: t => h
end

(d) fun (X Y Z : Type) (f : X->Y) (g : Y->Z) (a : X) =>
    g (f a)
5. (12 points) In this question, we’ll consider two different implementations of the same transformation on lists — one as an inductively defined relation and one as a Fixpoint.

(a) The relation \texttt{rdrop} is a three-place relation that holds between a number \( n \), a list \( xs \), and a list \( xs' \) if and only if \( xs' \) is the list obtained by dropping the first \( n \) elements of \( xs \). For example, the following are all provable instances of \texttt{rdrop}.

\[
\begin{align*}
\text{rdrop 3 [1,2,3,4,5]} & \Rightarrow [4,5] \\
\text{rdrop 2 [5,4,3,2,1]} & \Rightarrow [3,2,1] \\
\text{rdrop 5 [1,2,3]} & \Rightarrow []
\end{align*}
\]

Complete the following definition of \texttt{rdrop}.

\texttt{Inductive rdrop \{X : Type\} : nat \to list X \to list X \to Prop :=}

(b) Similarly, \texttt{fdrop} is a function that takes a number \( n \) and a list \( xs \) and returns the list consisting of all except the first \( n \) the elements of \( xs \). For example:

\[
\begin{align*}
\text{fdrop 3 [1,2,3,4,5]} & = [4,5]. \\
\text{fdrop 2 [5,4,3,2,1]} & = [3,2,1]. \\
\text{fdrop 5 [1,2,3]} & = []
\end{align*}
\]

Complete the following Fixpoint definition of \texttt{fdrop}.

\texttt{Fixpoint fdrop \{X : Type\} (n : nat) (xs : list X) : list X :=}
6. (20 points) Recall the definition of \texttt{beq_nat}:

\begin{verbatim}
Fixpoint beq_nat (n m : nat) : bool :=
  match n with
  | O => match m with
    | O => true
    | S m' => false
    end
  | S n' => match m with
    | O => false
    | S m' => beq_nat n' m'
    end
  end.
\end{verbatim}

Write out a careful informal proof of the following theorem, using the pedantic “template” style discussed in the notes. Make sure to state the induction hypothesis explicitly.

\textbf{Theorem:} For all natural numbers \( n \) and \( m \), if \( \texttt{beq_nat} n m = \texttt{true} \) then \( n = m \).

\textbf{Proof:}
7. (10 points) Recall the inductive definitions of logical conjunction and the property \texttt{beautiful}:

\[
\begin{align*}
\text{Inductive and} & \quad (P \ Q : \text{Prop}) : \text{Prop} := \\
\quad \text{conj} & \quad P \to Q \to (\text{and} P \ Q).
\end{align*}
\]

\text{Notation} "P \land Q" := (\text{and} P \ Q) : \text{type\_scope}.

\[
\begin{align*}
\text{Inductive beautiful} & \quad \text{nat} \to \text{Prop} := \\
\quad \text{b}_0 & \quad \text{beautiful} 0 \\
| \quad \text{b}_3 & \quad \text{beautiful} 3 \\
| \quad \text{b}_5 & \quad \text{beautiful} 5 \\
| \quad \text{b\_sum} & \quad \text{forall} n \ m, \ \text{beautiful} n \to \text{beautiful} m \to \text{beautiful} (n+m).
\end{align*}
\]

Suppose we have already proved the following theorem:

\text{Theorem b1000: beautiful} 1000.

Give a proof object for the following proposition. Show all parts of the proof object explicitly (i.e., do not use \_ anywhere).

\[
\begin{align*}
\text{Definition b\_facts} & \quad \text{forall} x, \\
\quad \text{beautiful} x \to \\
\quad (\text{beautiful} (1000 + x) \land \text{beautiful} 3) := 
\end{align*}
\]

8. (2 points) How many different proof objects are there for the proposition in the previous question?
9. (8 points) Recall the definition of existential quantification:

\[
\text{Inductive } \text{ex} (X : \text{Type}) (P : X \to \text{Prop}) : \text{Prop} := \\
\quad \text{ex_intro} : \forall \text{witness : } X, \ P \text{ witness } \to \text{ex } X \ P.
\]

(a) Write a proposition capturing the claim “there is some number whose successor is beautiful.”

(b) Give a proof object for this proposition.