Ph.D. Qualifying Examination
Principles of Programming Languages

Department of Computer Science
University of Wyoming

11 April 2005

Name: ___________________________________________

Instructions: There are three questions, if you choose to answer any of the three, be sure to answer all parts to that question. You may use Schmidt as a reference.
1. [Untyped λ-calculus and Fixed-point Combinators (2 parts)]

1a.) Consider the fixedpoint combinator $Y$ where

$$Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$$

Prove\(^1\) that $Y$ has the fixedpoint property, i.e. show that for all lambda terms $M$ the following holds:

$$YM = _\beta M(YM)$$

---

\(^1\)You will need to use $\beta$-conversion; e.g. use the equality $(\lambda x. M)N = M[x := N]$, possibly in both directions, to show that $YM = M(YM)$.  

Consider the untyped lambda terms (Λ) extended to include Booleans (B = \{true, false\}), constants for each integer (Z), an if-then-else operation, addition and subtraction on integers, and an equality test on integers.

\[
\Lambda ::= x | \lambda x. M | (MN) | B | \text{if } B \text{ then } M \text{ else } N | M + N | M - N | I \leq J
\]

where \(M, N \in \Lambda\) are lambda terms,
\(x \in \text{Var}\) is a variable,
\(M + N\) denotes ordinary addition on the integers, and
\(M - N\) denotes ordinary subtraction on the integers, and
\(I \leq J\) denotes the ordinary ordering in \(\mathbb{Z}\) when \(I, J \in \mathbb{Z}\).

The rules for evaluating if-then-else are

\[
\begin{align*}
\text{if true then } M \text{ else } N & \rightarrow M \\
\text{if false then } M \text{ else } N & \rightarrow N
\end{align*}
\]

Note that \(Y \in \Lambda\).

Use the fix-point property of \(Y\) (defined above in part a) to define a closed term in \(\Lambda\) implementing the following recursive description of the a summation operator:

\[
\text{sum } n \overset{\text{def}}{=} \text{if } n \leq 0 \text{ then } 0 \text{ else } n + \text{sum } (n - 1)
\]
2. [Simply Typed λ-calculus (2 parts)] Consider a simply typed lambda calculus defined as follows.

$$\Lambda ::= X \mid \lambda X : \theta. M \mid (MN)$$

where $X \in \text{Var}$ is a variable, and $M, N \in \Lambda$ are lambda terms.

Types are defined by the following grammar:

$$\theta ::= B \mid Z \mid \theta_1 \rightarrow \theta_2$$

A type assignment $\pi$ is a set of pairs $\{X_j : \theta_j\}_{0 \leq j < k}$ where $X_j$ is a variable and $\theta_j$ is a type. We define a special union operator on type assignments as follows:

$$\pi_1 \cup \pi_2 = \pi_2 \cup (\pi_1 - \{(X : \theta) \in \pi_1 | \exists \theta_1. (X : \theta_1) \in \pi_2\})$$

The following are the typing rules for this system.

$$\frac{}{\pi \vdash X \in \theta} \quad \text{if } (X : \theta) \in \pi$$

$$\frac{\pi \vdash M \in \theta_1 \rightarrow \theta_2 \quad \pi \vdash N \in \theta_1}{\pi \vdash M(N) \in \theta_2}$$

$$\frac{\pi \cup \{x : \theta_1\} \vdash M \in \theta_2}{\pi \vdash (\lambda x : \theta_1. M) \in \theta_1 \rightarrow \theta_2}$$
2a.) Use the typing rules to derive a type for the following term

\[(\lambda f : (\mathbb{Z} \to \mathbb{B}) \to \mathbb{Z}. (\lambda g : \mathbb{Z} \to \mathbb{B}. (\lambda h : \mathbb{B}. f(g))))\]

2b.) Use the typing rules to derive a type for the following term

\[(\lambda y : \mathbb{Z} \to \mathbb{Z} \to \mathbb{B}. (\lambda x : \mathbb{Z}. (\lambda z : \mathbb{Z}. (y x) z)))\]
3. [Denotational Semantics (2 Parts)] Use the semantic equations from Schmidt (pg. 13,14) to prove the following phrases are equivalent for all stores.

3a.) $\sem{\text{if } E \text{ then } C_1 \text{ else } C_2 \text{ fi}; C_3} = \sem{\text{if } E \text{ then } C_1; C_3 \text{ else } C_2; C_3 \text{ fi}}$

---

$^2$You should realize that both expressions denote functions of type $\mathsf{Store} \rightarrow \mathsf{Store}_{\bot}$ and so should use extensionality.
3b.) The following identities hold for all stores and all commands $C$.

i.) $\llbracket \text{while } (0 = 0) \text{ do skip od : comm} \rrbracket (s) = \bot$

ii.) $\llbracket C : \text{comm} \rrbracket (\bot) = \bot$

Use these two facts to show the following equivalence holds.

$\llbracket \text{while } (0 = 0) \text{ do skip od : } C_1 : \text{comm} \rrbracket = \llbracket C_1 ; \text{while } (0 = 0) \text{ do skip od : comm} \rrbracket$