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 β-conversion; *e.g.* use the equality $(\lambda x.M)N = M[x := N]$, possibly in both directions, to show that $YM = M(YM)$.

1b.) Consider the untyped lambda terms (Λ) extended to include Booleans ($\mathbb{B} = \{true, false\}$), constants for each integer ($\mathbb{Z}$), an `if-then-else` operation, addition and subtraction on integers, and an equality test on integers.

$$\Lambda ::= x|\lambda x.M|(MN)|\mathbb{B}|\mathbb{Z}|\texttt{if } B \texttt{ then } M \texttt{ else } N|M+N|M-N|I \leq J$$

where $M, N \in \Lambda$ are lambda terms,
$x \in \mathbf{Var}$ 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

$$\texttt{if } true \texttt{ then } M \texttt{ else } N \rightarrow M$$
$$\texttt{if } false \texttt{ then } M \texttt{ else } N \rightarrow N$$

Note that $Y \in \Lambda$.

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

$\texttt{sum n} \stackrel{\text{def}}{=} \texttt{if n} \leq \texttt{0 then 0 else n + sum (n - 1)}$

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

Types are defined by the following grammar:

$$\theta ::= \mathbb{B} \mid \mathbb{Z} \mid \theta_1 \to \theta_2$$

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

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

The following are the typing rules for this system.

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

$$\frac{\pi \vdash M \in \theta_1 \to \theta_2 \qquad \pi \vdash N \in \theta_1}{\pi \vdash M(N) \in \theta_2}$$

$$\frac{\pi \uplus \{x : \theta_1\} \quad \vdash \quad M \in \theta_2}{\pi \quad \vdash \quad (\lambda x : \theta_1 . M) \in \theta_1 \to \theta_2}$$

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

$$(\lambda y\!:\!\mathbb{Z} \to \mathbb{Z} \to \mathbb{B}.\,(\lambda x\!:\!\mathbb{Z}.(\lambda z : \mathbb{Z}.\,(yx)z)))$$

3. [*Denotational Semantics* (2 Parts)] Use the semantic equations from Schmidt (pg. 13,14) to prove[2] the following phrases are equivalent for all stores.

3a.) $[\![\mathbf{if}\ E\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \mathbf{fi}; C_3]\!] = [\![\mathbf{if}\ E\ \mathbf{then}\ C_1; C_3\ \mathbf{else}\ C_2; C_3\ \mathbf{fi}]\!]$

[2] You should realize that both expressions denote functions of type $Store \rightarrow Store_\perp$ and so should use extensionality.

3b.) The following identities hold for all stores and all commands C.

$$\begin{array}{ll} i.) & [\![\textbf{while}\,(0=0)\,\textbf{do skip od} : comm]\!](s) = \bot \\ ii.) & [\![C : comm]\!](\bot) = \bot \end{array}$$

Use these two facts to show the following equivalence holds.

$$[\![\textbf{while}\,(0=0)\,\textbf{do skip od}; C_1 : comm]\!] = [\![C_1; \textbf{while}\,(0=0)\,\textbf{do skip od} : comm]\!]$$