Indian Institute of Technology, Delhi
Department of Computer Science and Engineering

CS 232 N	Programming Languages	Minor II
March 15, 2000	16:00-17:00	Maximum Marks: 80

There are several varieties of λ-calculi — in this exam we will focus on a pure, simply-typed λ-calculus. Assuming a denumerable set of variables $\mathcal{X}$, the syntactic category of expressions is given inductively by the abstract grammar:

$$e \quad ::= \quad x \mid (\lambda x.e_1) \mid (e_1\ e_2)$$

Expressions have types drawn from a set **Types** that has the following inductive characterization:

$$\tau \quad ::= \quad \alpha \ \text{(type variable)} \mid \tau_1 \to \tau_2 \ \text{(function types)}$$

The structural **static semantics** or **typing rules** are:

$$\Gamma \vdash x : \tau \ \text{ if } \tau = \Gamma(x) \qquad \frac{\Gamma[x \mapsto \tau_1] \vdash e_1 : \tau_2}{\Gamma \vdash (\lambda x.e_1) : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash (e_1\ e_2) : \tau_2}$$

where $\Gamma \in \mathcal{X} \longrightarrow_{fin}$ **Types** is called a type environment.

Q1 [5+10+1 Marks] **Typing.** (i) If β-equal terms must have the same types, what is the type of a combinator Y satisfying $(Y\ f)\ =_\beta\ f(Y\ f)$?
Y : $\underline{(\gamma \to \gamma) \to \gamma}$.
(let $f : \alpha, (Y\ f) : \beta$, then $Y : \alpha \to \beta$ and $\alpha = \beta \to \beta$)

(ii) Show that the λ-expression $(\lambda x.(x\ x))$ is *not* typable under the above typing rules.
Let $x : \alpha, (x\ x) : \beta$, then $x : \alpha \to \beta$, which *fails to unify* with α.

From this example, what can we conclude about the typability of the following λ-calculus expressions? *Not typable, because they contain self-application* $(x\ x)$
$\Omega \ \equiv\ ((\lambda x.(xx))(\lambda x.(xx)))$ and $Y_{Curry} \ \equiv\ \lambda f.((\lambda x.f(xx))(\lambda x.f(xx)))$

Q2 [8+16 marks] **Reduction Semantics.** "Call by lazy" is an evaluation strategy for the λ-calculus that is defined by the following evaluation rules:

$$((\lambda x.e_1)\ e_2) \rhd_1 e_1[e_2/x] \qquad \frac{e_1 \rhd_1 e_1'}{(e_1\ e_2) \rhd_1 (e_1'\ e_2)}$$

(a) Characterize the normal forms (called weak head normal forms) under this strategy?

$$b \ ::= \ x \mid (\lambda x.e) \mid x\ e_1\ e_2\ \ldots\ e_k$$

(b) By the Principle of <u>Qualification</u>, we decide to extend our λ-calculus with a construct for local definitions: **let** $x \stackrel{\triangle}{=} e_1$ **in** e_2 **end**. Extend the typing rules and the reduction semantics of the call-by-lazy λ-calculus to deal with this new construct. Which principle guides the formulation of your new rule? <u>Principle of Correspondence</u>.

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma[x \mapsto \tau_1] \vdash e_2 : \tau}{\Gamma \vdash \mathbf{let}\ x \stackrel{\triangle}{=} e_1\ \mathbf{in}\ e_2\ \mathbf{end} : \tau} \qquad \mathbf{let}\ x \stackrel{\triangle}{=} e_1\ \mathbf{in}\ e_2\ \mathbf{end} \rhd_1 e_2[e_1/x]$$

Q3 [40 Marks] **Subject Reduction.** An important theorem about the typed λ-calculi is the "subject reduction" theorem which says that the type of an expression does not change during evaluation. A consequence is that if an expression type-checks (at compile time) one does not require type checking at run-time.

Assume lemmas that say (i) if $\Gamma'[y \mapsto \tau_1'] \vdash e' : \tau'$ then for fresh variable $z \notin dom(\Gamma')$ $\Gamma'[z \mapsto \tau_1'] \vdash e'[z/y] : \tau'$ and (ii) if $\Gamma'[y \mapsto \tau_1'] \vdash e' : \tau'$ and $y \notin FV(e')$, then $\Gamma' \vdash e' : \tau'$.

Prove the following proposition that types are preserved under substitution:

$$\text{If } \Gamma[x \mapsto \tau_1] \vdash e : \tau \text{ and } \Gamma \vdash e_1 : \tau_1, \text{ then } \Gamma \vdash e[e_1/x] : \tau$$

PROOF. (by induction on length of e.)
Base cases: (i) $e \equiv x$: then $e[e_1/x] \equiv e_1$ and clearly $\Gamma \vdash e_1 : \tau$ (given).
(ii) $e \equiv y \not\equiv x$: then $e[e_1/x] \equiv y$ and applying Lemma (ii) to $\Gamma[x \mapsto \tau_1] \vdash y : \tau$ (given), we have $\Gamma \vdash y : \tau$.

Induction Hypothesis: Assume that for all Γ', τ' and all e' of length $\leq n$, we have shown that if $\Gamma'[x \mapsto \tau_1] \vdash e' : \tau'$ and $\Gamma' \vdash e_1 : \tau_1$, then $\Gamma' \vdash e'[e_1/x] : \tau'$. (*Note that the generalization to all Γ', τ', e' is absolutely necessary for the proof to work.*)

Induction Steps Consider e of length $n+1$.
(iii) $e \equiv (e_1' \; e_2')$, where e_1', e_2' are if length $\leq n$: Clearly, we must have $\Gamma[x \mapsto \tau_1] \vdash e_1' : \tau_2 \to \tau$ and $\Gamma[x \mapsto \tau_1] \vdash e_2' : \tau_2$ for some τ_2. By the induction hypothesis applied on each of these, we have $\Gamma \vdash e_1'[e_1/x] : \tau_2 \to \tau$ and $\Gamma \vdash e_2'[e_1/x] : \tau_2$. So $\Gamma \vdash (e_1'[e_1/x] \; e_2'[e_1/x]) : \tau$, but $(e_1'[e_1/x] \; e_2'[e_1/x]) \equiv (e_1' \; e_2')[e_1/x]$.

(iv) $e \equiv (\lambda x.e_1')$: Note that $(\lambda x.e_1')[e_1/x] \equiv (\lambda x.e_1')$, and $x \notin FV(e)$, so applying Lemma (ii) to $\Gamma[x \mapsto \tau_1] \vdash e : \tau$ (given), we have $\Gamma \vdash e[e_1/x] : \tau$.

(v) $e \equiv (\lambda y.e_1')$, $x \not\equiv y$, where e_1' is of length $\leq n$. From $\Gamma[x \mapsto \tau_1] \vdash (\lambda y.e_1') : \tau$ (given), we must have $\Gamma[x \mapsto \tau_1][y \mapsto \tau_2] \vdash e_1' : \tau_3$ for some τ_2, τ_3 ($\tau = \tau_2 \to \tau_3$).

There are two subcases.
If $x \notin FV(e_1')$ or $y \notin FV(e_1)$, then $(\lambda y.e_1')[e_1/x] \equiv \lambda y.(e_1'[e_1/x])$. Since $\Gamma[x \mapsto \tau_1][y \mapsto \tau_2] = \Gamma[y \mapsto \tau_2][x \mapsto \tau_1]$, by the IH, we have $\Gamma[y \mapsto \tau_2] \vdash e_1'[e_1/x] : \tau_3$, and so we get $\Gamma \vdash (\lambda y.e_1')[e_1/x] : \tau_2 \to \tau_3$.

If $x \in FV(e_1')$ and $y \in FV(e_1)$, then $(\lambda y.e_1')[e_1/x] \equiv \lambda z.(e_1'[z/y][e_1/x])$ for some fresh z. Applying Lemma (i) we get $\Gamma[x \mapsto \tau_1][z \mapsto \tau_2] \vdash e_1'[z/y] : \tau_3$. Now, $e_1'[z/y]$ is also of length $\leq n$, so by the IH we get $\Gamma[z \mapsto \tau_2] \vdash e_1'[z/y][e_1/x] : \tau_3$, from which $\Gamma \vdash (\lambda y.e_1')[e_1/x] : \tau_2 \to \tau_3$ follows immediately.