Name:____

CSE 505, Fall 2005, Midterm Examination
8 November 2005

Please do not turn the page until everyone is ready.

Rules:

- The exam is closed-book, closed-note, except for one side of one 8.5x11in piece of paper.
- **Please stop promptly at 1:20.**
- You can rip apart the pages, but please write your name on each page.
- There are **140 points** total, distributed **unevenly** among 6 questions (which have multiple parts).

Advice:

- Read questions carefully. Understand a question before you start writing.
- Write down thoughts and intermediate steps so you can get partial credit.
- The questions are not necessarily in order of difficulty. **Skip around.** In particular, do not spend so much time on a proof that you do not get to all the problems.
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

For your reference:

$$
\begin{array}{rcl}
s & ::= & \mathsf{skip} \mid x := e \mid s; s \mid \mathsf{if}\ e\ s\ s \mid \mathsf{while}\ e\ s \\
e & ::= & c \mid x \mid e + e \mid e * e \\
(c & \in & \{\ldots, -2, -1, 0, 1, 2, \ldots\}) \\
(x & \in & \{\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{y}_1, \mathtt{y}_2, \ldots, \mathtt{z}_1, \mathtt{z}_2, \ldots, \ldots\})
\end{array}
$$

$\boxed{H\ ;\ e\ \Downarrow\ c}$

$$
\textsc{const}\ \frac{}{H\ ;\ c\ \Downarrow\ c}
\qquad
\textsc{var}\ \frac{}{H\ ;\ x\ \Downarrow\ H(x)}
\qquad
\textsc{add}\ \frac{H\ ;\ e_1\ \Downarrow\ c_1 \qquad H\ ;\ e_2\ \Downarrow\ c_2}{H\ ;\ e_1 + e_2\ \Downarrow\ c_1 + c_2}
\qquad
\textsc{mult}\ \frac{H\ ;\ e_1\ \Downarrow\ c_1 \qquad H\ ;\ e_2\ \Downarrow\ c_2}{H\ ;\ e_1 * e_2\ \Downarrow\ c_1 * c_2}
$$

$\boxed{H_1\ ;\ s_1\ \rightarrow\ H_2\ ;\ s_2}$

$$
\textsc{assign}\ \frac{H\ ;\ e\ \Downarrow\ c}{H\ ;\ x := e\ \rightarrow\ H, x \mapsto c\ ;\ \mathsf{skip}}
\qquad
\textsc{seq1}\ \frac{}{H\ ;\ \mathsf{skip}; s\ \rightarrow\ H\ ;\ s}
\qquad
\textsc{seq2}\ \frac{H\ ;\ s_1\ \rightarrow\ H'\ ;\ s_1'}{H\ ;\ s_1; s_2\ \rightarrow\ H'\ ;\ s_1'; s_2}
$$

$$
\textsc{if1}\ \frac{H\ ;\ e\ \Downarrow\ c \qquad c > 0}{H\ ;\ \mathsf{if}\ e\ s_1\ s_2\ \rightarrow\ H\ ;\ s_1}
\qquad
\textsc{if2}\ \frac{H\ ;\ e\ \Downarrow\ c \qquad c \leq 0}{H\ ;\ \mathsf{if}\ e\ s_1\ s_2\ \rightarrow\ H\ ;\ s_2}
$$

$$
\begin{array}{rcl}
e & ::= & \lambda x.\ e \mid x \mid e\ e \mid c \\
v & ::= & \lambda x.\ e \mid c \\
\tau & ::= & \mathsf{int} \mid \tau \rightarrow \tau
\end{array}
$$

$\boxed{e \rightarrow e'}$

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

$\boxed{e[e'/x] = e''}$

$$
\frac{}{x[e/x] = e}
\qquad
\frac{e_1[e/x] = e_1' \qquad y \neq x \qquad y \notin FV(e)}{(\lambda y.\ e_1)[e/x] = \lambda y.\ e_1'}
$$

$$
\frac{y \neq x}{y[e/x] = y}
\qquad
\frac{e_1[e/x] = e_1' \qquad e_2[e/x] = e_2'}{(e_1\ e_2)[e/x] = e_1'\ e_2'}
$$

$\boxed{\Gamma \vdash e : \tau}$

$$
\frac{}{\Gamma \vdash c : \mathsf{int}}
\qquad
\frac{}{\Gamma \vdash x : \Gamma(x)}
\qquad
\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.\ e : \tau_1 \rightarrow \tau_2}
\qquad
\frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}
$$

Name:____

1. (IMP with choice)

   (a) (**10** points) Let "?" be a choice operator for IMP expressions: $e_1?e_2$ chooses either e_1 or e_2 and evaluates its choice to produce an answer. Give semantic rules for this extension.

   (b) (**20** points) Theorem: If e_1 is equivalent to e_2, then e_1 is equivalent to $e_1?e_2$.

   - Restate this theorem formally.
   - Prove this theorem formally.

Solution:

(a)

$$\textsc{left}\;\frac{H\;;\;e_1\;\Downarrow\;c}{H\;;\;e_1?e_2\;\Downarrow\;c}\qquad\qquad \textsc{right}\;\frac{H\;;\;e_2\;\Downarrow\;c}{H\;;\;e_1?e_2\;\Downarrow\;c}$$

(b) For all H, e_1, e_2, and c, suppose $H\;;\;e_1\;\Downarrow\;c$ if and only if $H\;;\;e_2\;\Downarrow\;c$. Then $H\;;\;e_1\;\Downarrow\;c$ if and only if $H\;;\;e_1?e_2\;\Downarrow\;c$.

We prove the two directions of the if-and-only-if separately. First assume $H\;;\;e_1\;\Downarrow\;c$. Then we can use LEFT to derive $H\;;\;e_1?e_2\;\Downarrow\;c$. Now assume $H\;;\;e_1?e_2\;\Downarrow\;c$. Then inverting the derivation ensures the derivation ends with either LEFT or RIGHT. If LEFT, then $H\;;\;e_1\;\Downarrow\;c$ directly. If RIGHT, then $H\;;\;e_2\;\Downarrow\;c$, but by assumption that means $H\;;\;e_1\;\Downarrow\;c$.

Name:____

2. (Bad statement rules)

   (a) (**10** points) Why do we not have this rule in our IMP statement semantics?

$$\frac{H\ ;\ s_1\ \rightarrow\ H'\ ;\ s_1'}{H\ ;\ s_1; (s_2; s_3)\ \rightarrow\ H'\ ;\ s_1'; (s_2; s_3)}$$

   (b) (**10** points) Why do we not have this rule in our IMP statement semantics?

$$\frac{H\ ;\ s_1\ \rightarrow\ H'\ ;\ s_1'}{H\ ;\ s_2; s_1\ \rightarrow\ H'\ ;\ s_2; s_1'}$$

Solution:

(a) It is unnecessary because we can use SEQ2 to conclude $H\ ;\ s_1; (s_2; s_3)\ \rightarrow\ H'\ ;\ s_1'; (s_2; s_3)$ given $H\ ;\ s_1\ \rightarrow\ H'\ ;\ s_1'$ – we just instantiate the s_2 in the rule with $s_2; s_3$.

(b) It is not what we "want" – the purpose of a sequence of statements is to execute the statements *in order*. This rule would make our language non-deterministic in a way we don't want because it lets us execute the two parts of a sequence in either order (or in fact we can interleave their execution in any way).

Name:____

3. (Functional programming)

(a) (**10** points) Consider this Caml code:

```
type t = A of int | B of (int->int)
let x = 2
let f y = x + y
let ans1 = (let x = 3 in
            let a = A (f 4) in
            let x = 5 in
            match a with A x -> x | B x -> x 6)
let ans2 = (let x = 3 in
            let b = B f in
            let x = 5 in
            match b with A x -> x | B x -> x 6)
```

After evaluating this code, what values are `ans1` and `ans2` bound to?

(b) (**10** points) Consider this Caml code:

```
let rec g x =
  match x with
    [] -> []
  | hd::tl -> (fun y -> hd + y)::(g tl)
```

i. What does this function do?

ii. What is this function's type?

iii. Write a function `h` that is the *inverse* of `g`. That is, `fun x -> h (g x)` would return a value equivalent to its input.

Solution:

(a) `ans1` is bound to 6 and `ans2` is bound to 8.

(b) This function takes a list of integers and returns a list of functions where the i^{th} element in the output list returns the sum of its input and the i^{th} element of the input list.

(c) `int list -> ((int -> int) list)`

(d) `let rec h x = match x with [] -> [] | hd::tl -> (hd 0)::(h tl)`

Name:____

4. (λ encodings) Recall this encoding of booleans in the λ-calculus:
   "true" $\lambda x.\ \lambda y.\ x$
   "false" $\lambda x.\ \lambda y.\ y$
   "if" $\lambda b.\ \lambda t.\ \lambda f.\ b\ t\ f$
   (a) (**10** points) Extend this encoding with a λ term that encodes *(inclusive) or*.
   (b) (**10** points) Extend this encoding with a λ term that encodes *not*.

Solution:

(a) "or" $\lambda b_1.\ \lambda b_2.\ b_1\ (\lambda x.\ \lambda y.\ x)\ b_2$
(b) "not" $\lambda b.\ b\ (\lambda x.\ \lambda y.\ y)\ (\lambda x.\ \lambda y.\ x)$

Name:____

5. (Simply-Typed λ calculus)
   For all subproblems, assume the simply-typed λ calculus.

   (a) (**6** points) Give a Γ, e_1, e_2, and τ such that $\Gamma \vdash e_1 : \tau$ and $\Gamma \vdash e_2 : \tau$ and $e_1 \neq e_2$.

   (b) (**6** points) Give a Γ_1, Γ_2, e, and τ such that $\Gamma_1 \vdash e : \tau$ and $\Gamma_2 \vdash e : \tau$ and $\Gamma_1 \neq \Gamma_2$.

   (c) (**8** points) Give a Γ, e, τ_1, and τ_2 such that $\Gamma \vdash e : \tau_1$ and $\Gamma \vdash e : \tau_2$ and $\tau_1 \neq \tau_2$.

   Solution:

   (a) $\Gamma = x{:}\mathsf{int}, y{:}\mathsf{int}$, $e_1 = x$, $e_2 = y$, $\tau = \mathsf{int}$.

   (b) $\Gamma_1 = x{:}\mathsf{int}$, $\Gamma_2 = x{:}\mathsf{int}, y{:}\mathsf{int}$, $e = x$, $\tau = \mathsf{int}$.

   (c) $\Gamma = \cdot$, $e = \lambda x.\ x$, $\tau_1 = \mathsf{int} \to \mathsf{int}$, $\tau_2 = (\mathsf{int} \to \mathsf{int}) \to (\mathsf{int} \to \mathsf{int})$

Name:____

6. (Type-Safety)
   We add an explicit infinite-loop to the simply-typed λ-calculus: The term ∞ simply "reduces to itself".
   (a) (**5** points) Extend the semantics of the call-by-value λ-calculus to include ∞.
   (b) (**10** points) Extend the type system of the simply-typed λ-calculus to include ∞. Be as permissive as possible considering the next problem.
   (c) (**15** points) Prove that your extensions maintain type safety. Do *not* repeate the entire type-safety proof. Rather, for each of these lemmas, remind us the structure of the proof (i.e., the induction hypothesis) and then prove any new cases introduced by your extensions.
   - Preservation: If $\cdot \vdash e : \tau$ and $e \to e'$, then $\cdot \vdash e' : \tau$.
   - Progress: If $\cdot \vdash e : \tau$, then e is a value or there exists an e' such that $e \to e'$.
   - Substitution: If $\Gamma, x{:}\tau' \vdash e_1 : \tau$ and $\Gamma \vdash e_2 : \tau'$, then $\Gamma \vdash e_1[e_2/x] : \tau$.

Solution:

(a)

$$\frac{}{\infty \to \infty}\ \textsc{inf}$$

$$\frac{}{\infty[e/x] = \infty}$$

(b)

$$\frac{}{\Gamma \vdash \infty : \tau}$$

(c)
- Preservation: By induction on the (height of the) derivation that $e \to e'$. The new case is that the derivation ends with INF. But then e' is ∞ so we can use our new typing rule to conclude $\cdot \vdash e' : \tau$.
- Progress: By induction on the structure (height) of expressions. The new case is that e is ∞, in which case we can use INF to take a step.
- Substitution: By induction on the typing derivation of e_1. The new case is $e_1 = \infty$, in which case $e_1[e_2/x] = \infty$, so we can use our new typing rule to derive $\Gamma \vdash e_1[e_2/x] : \tau$.