Name:____

CSE 505, Fall 2005, Final Examination
15 December 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 12:20.**
- You can rip apart the pages, but please write your name on each page.
- There are **120 points** total, distributed **evenly** among 6 questions (most of 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.**
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

For your reference (page 1 of 2):

$$\begin{array}{rcl} e & ::= & \lambda x.\ e \mid x \mid e\ e \mid c \mid \{l_1 = e_1, \ldots, l_n = e_n\} \mid e.l_i \mid \mathsf{fix}\ e \\ v & ::= & \lambda x.\ e \mid c \mid \{l_1 = v_1, \ldots, l_n = v_n\} \\ \tau & ::= & \mathsf{int} \mid \tau \to \tau \mid \{l_1 : \tau_1, \ldots, l_n : \tau_n\} \end{array}$$

$\boxed{e \to e' \text{ and } \Gamma \vdash e : \tau \text{ and } \tau_1 \leq \tau_2}$

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

$$\frac{}{\{l_1 = v_1, \ldots, l_n = v_n\}.l_i \to v_i}$$

$$\frac{e_i \to e_i'}{\{l_1 = v_1, \ldots, l_{i-1} = v_{i-1}, l_i = e_i, \ldots, l_n = e_n\} \to \{l_1 = v_1, \ldots, l_{i-1} = v_{i-1}, l_i = e_i', \ldots, l_n = e_n\}}$$

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

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \ldots \qquad \Gamma \vdash e_n : \tau_n \qquad \text{labels distinct}}{\Gamma \vdash \{l_1 = e_1, \ldots, l_n = e_n\} : \{l_1 : \tau_1, \ldots, l_n : \tau_n\}} \qquad \frac{\Gamma \vdash e : \{l_1 : \tau_1, \ldots, l_n : \tau_n\} \qquad 1 \leq i \leq n}{\Gamma \vdash e.l_i : \tau_i}$$

$$\frac{\Gamma \vdash e : \tau \qquad \tau \leq \tau'}{\Gamma \vdash e : \tau'}$$

$$\frac{}{\{l_1{:}\tau_1, \ldots, l_n{:}\tau_n, l{:}\tau\} \leq \{l_1{:}\tau_1, \ldots, l_n{:}\tau_n\}}$$

$$\frac{}{\{l_1{:}\tau_1, \ldots, l_{i-1}{:}\tau_{i-1}, l_i{:}\tau_i, \ldots, l_n{:}\tau_n\} \leq \{l_1{:}\tau_1, \ldots, l_i{:}\tau_i, l_{i-1}{:}\tau_{i-1}, \ldots, l_n{:}\tau_n\}}$$

$$\frac{\tau_i \leq \tau_i'}{\{l_1{:}\tau_1, \ldots, l_i{:}\tau_i, \ldots, l_n{:}\tau_n\} \leq \{l_1{:}\tau_1, \ldots, l_i{:}\tau_i', \ldots, l_n{:}\tau_n\}} \qquad \frac{\tau_3 \leq \tau_1 \qquad \tau_2 \leq \tau_4}{\tau_1 \to \tau_2 \leq \tau_3 \to \tau_4} \qquad \frac{}{\tau \leq \tau}$$

$$\begin{array}{rcl} e & ::= & c \mid x \mid \lambda x{:}\tau.\ e \mid e\ e \mid \Lambda\alpha.\ e \mid e[\tau] \\ \tau & ::= & \mathsf{int} \mid \tau \to \tau \mid \alpha \mid \forall \alpha.\tau \\ v & ::= & c \mid \lambda x{:}\tau.\ e \mid \Lambda\alpha.\ e \end{array} \qquad \begin{array}{rcl} \Gamma & ::= & \cdot \mid \Gamma, x{:}\tau \\ \Delta & ::= & \cdot \mid \Delta, \alpha \end{array}$$

$\boxed{e \to e' \text{ and } \Delta; \Gamma \vdash e : \tau}$

$$\frac{e \to e'}{e\ e_2 \to e'\ e_2} \qquad \frac{e \to e'}{v\ e \to v\ e'} \qquad \frac{e \to e'}{e[\tau] \to e'[\tau]} \qquad \frac{}{(\lambda x{:}\tau.\ e)v \to e[v/x]} \qquad \frac{}{(\Lambda\alpha.\ e)[\tau] \to e[\tau/\alpha]}$$

$$\frac{}{\Delta; \Gamma \vdash x : \Gamma(x)} \qquad \frac{}{\Delta; \Gamma \vdash c : \mathsf{int}} \qquad \frac{\Delta; \Gamma, x{:}\tau_1 \vdash e : \tau_2 \qquad \Delta \vdash \tau_1}{\Delta; \Gamma \vdash \lambda x{:}\tau_1.\ e : \tau_1 \to \tau_2} \qquad \frac{\Delta, \alpha; \Gamma \vdash e : \tau_1}{\Delta; \Gamma \vdash \Lambda\alpha.\ e : \forall \alpha.\tau_1}$$

$$\frac{\Delta; \Gamma \vdash e_1 : \tau_2 \to \tau_1 \qquad \Delta; \Gamma \vdash e_2 : \tau_2}{\Delta; \Gamma \vdash e_1\ e_2 : \tau_1} \qquad \frac{\Delta; \Gamma \vdash e : \forall \alpha.\tau_1 \qquad \Delta \vdash \tau_2}{\Delta; \Gamma \vdash e[\tau_2] : \tau_1[\tau_2/\alpha]}$$

Name:____

$$
\begin{array}{rcl}
e & ::= & c \mid \lambda x.\ e \mid e\ e \mid (e, e) \mid e.1 \mid e.2 \mid \mathsf{letcc}\ x.\ e \mid \mathsf{throw}\ e\ e \mid \mathsf{continuation}\ E \\
E & ::= & [\cdot] \mid E\ e \mid v\ E \mid (E, e) \mid (v, E) \mid E.1 \mid E.2 \mid \mathsf{throw}\ E\ e \mid \mathsf{throw}\ v\ E \\
v & ::= & c \mid \lambda x.\ e \mid (v, v) \mid \mathsf{continuation}\ E
\end{array}
$$

$$
\frac{}{(\lambda x.\ e)\ v \xrightarrow{\text{p}} e[v/x]} \qquad \frac{}{(v_1, v_2).1 \xrightarrow{\text{p}} v_1} \qquad \frac{}{(v_1, v_2).2 \xrightarrow{\text{p}} v_2}
$$

$$
\frac{e \xrightarrow{\text{p}} e'}{E[e] \to E[e']} \qquad \frac{}{E[\mathsf{letcc}\ x.\ e] \to E[e[\mathsf{continuation}\ E/x]]} \qquad \frac{}{E[\mathsf{throw}\ (\mathsf{continuation}\ E')\ v] \to E'[v]}
$$

$$
\begin{array}{rcl}
e & ::= & \ldots \mid \mathsf{inl}(e) \mid \mathsf{inr}(e) \mid (\mathsf{case}\ e\ x.e \mid x.e) \mid \mathsf{roll}_\tau\ e \mid \mathsf{unroll}\ e \mid \mathsf{raise}\ e \mid \mathsf{try}\ e\ \mathsf{catch}\ (c)\ e \\
\tau & ::= & \ldots \mid \tau_1 + \tau_2 \mid \mu\alpha.\tau \\
v & ::= & \ldots \mid \mathsf{inl}(v) \mid \mathsf{inr}(v) \mid \mathsf{roll}_\tau\ v
\end{array}
$$

$$
\frac{}{\mathsf{case}\ \mathsf{inl}(v)\ x.e_1 \mid x.e_2 \to e_1[v/x]} \qquad \frac{}{\mathsf{case}\ \mathsf{inr}(v)\ x.e_1 \mid x.e_2 \to e_2[v/x]} \qquad \frac{e \to e'}{\mathsf{inl}(e) \to \mathsf{inl}(e')}
$$

$$
\frac{e \to e'}{\mathsf{inr}(e) \to \mathsf{inr}(e')} \qquad \frac{e \to e'}{\mathsf{case}\ e\ x.e_1 \mid x.e_2 \to \mathsf{case}\ e'\ x.e_1 \mid x.e_2} \qquad \frac{e \to e'}{\mathsf{roll}_{\mu\alpha.\tau}\ e \to \mathsf{roll}_{\mu\alpha\tau}\ e'}
$$

$$
\frac{e \to e'}{\mathsf{unroll}\ e \to \mathsf{unroll}\ e'} \qquad \frac{}{\mathsf{unroll}\ (\mathsf{roll}_{\mu\alpha.\tau}\ v) \to v} \qquad \frac{e \to e'}{\mathsf{raise}\ e \to \mathsf{raise}\ e'}
$$

$$
\frac{e_1 \to e_1'}{\mathsf{try}\ e_1\ \mathsf{catch}\ (c)\ e_2 \to \mathsf{try}\ e_1'\ \mathsf{catch}\ (c)\ e_2} \qquad \frac{}{\mathsf{try}\ v\ \mathsf{catch}\ (c)\ e_2 \to v} \qquad \frac{}{\mathsf{try}\ \mathsf{raise}\ c\ \mathsf{catch}\ (c)\ e_2 \to e_2}
$$

$$
\frac{c \neq c'}{\mathsf{try}\ \mathsf{raise}\ c'\ \mathsf{catch}\ (c)\ e_2 \to \mathsf{raise}\ c'}
$$

many "bubble up exception" rules omitted

$$
\frac{\Delta; \Gamma \vdash e : \tau_1}{\Gamma \vdash \mathsf{inl}(e) : \tau_1 + \tau_2} \qquad \frac{\Delta; \Gamma \vdash e : \tau_2}{\Gamma \vdash \mathsf{inr}(e) : \tau_1 + \tau_2} \qquad \frac{\Delta; \Gamma \vdash e : \tau_1 + \tau_2 \qquad \Delta; \Gamma, x{:}\tau_1 \vdash e_1 : \tau \qquad \Delta; \Gamma, x{:}\tau_2 \vdash e_2 : \tau}{\Delta; \Gamma \vdash \mathsf{case}\ e\ x.e_1 \mid x.e_2 : \tau}
$$

$$
\frac{\Delta; \Gamma \vdash e : \tau[(\mu\alpha.\tau)/\alpha]}{\Delta; \Gamma \vdash \mathsf{roll}_{\mu\alpha.\tau}\ e : \mu\alpha.\tau} \qquad \frac{\Delta; \Gamma \vdash e : \mu\alpha.\tau}{\Delta; \Gamma \vdash \mathsf{unroll}\ e : \tau[(\mu\alpha.\tau)/\alpha]}
$$

$$
\frac{\Delta; \Gamma \vdash e : \mathsf{int} \qquad \Delta \vdash \tau}{\Delta; \Gamma \vdash \mathsf{raise}\ e : \tau} \qquad \frac{\Delta; \Gamma \vdash e_1 : \tau \qquad \Delta; \Gamma \vdash e_2 : \tau}{\Delta; \Gamma \vdash \mathsf{try}\ e_1\ \mathsf{catch}\ (c)\ e_2 : \tau}
$$

$$
e ::= \ldots \mid \mathsf{ref}\ e \mid !e \mid e1 := e2 \mid r \qquad \tau ::= \ldots \mid \tau\ \mathsf{ref} \qquad v ::= \ldots \mid r \qquad H ::= \cdot \mid H, r \mapsto v
$$

$\boxed{(H; e) \to (H'; e') \text{ and } \Delta; \Gamma \vdash e : \tau}$

$$
\frac{}{H; (\lambda x.\ e)\ v \to (H; e[v/x])} \qquad \frac{H; e_1 \to H'; e_1'}{H; e_1\ e_2 \to (H'; e_1'\ e_2)} \qquad \frac{r \notin \mathrm{Dom}(H)}{(H; \mathsf{ref}\ v) \to (H, r \mapsto v; r)}
$$

$$
\frac{}{(H; !r) \to (H; H(r))} \qquad \frac{}{(H; r := v) \to (H, r \mapsto v; ())}
$$

many inductive rules omitted

$$
\frac{\Delta; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \mathsf{ref}\ e : \tau\ \mathsf{ref}} \qquad \frac{\Delta; \Gamma \vdash e : \tau\ \mathsf{ref}}{\Delta; \Gamma \vdash !e : \tau} \qquad \frac{\Delta; \Gamma \vdash e_1 : \tau\ \mathsf{ref} \qquad \Delta; \Gamma \vdash e_2 : \tau}{\Delta; \Gamma \vdash e_1 := e_2 : \mathsf{unit}}
$$

Name:____

1. Here are two type definitions for different representations of linked lists of integers:

   - type t1 = $\mu\alpha.(\mathsf{unit} + (\mathsf{int} * \alpha))$
   - type t2 = $\mu\alpha.((\alpha * \mathsf{int}) + \mathsf{unit})$

   Write a typed λ-calculus program of the form $\mathsf{fix}(\lambda \mathit{convert} : \_.\ \lambda \mathit{lst} : \_.\ \_\_\_\_\_\_\_\_\_)$ for converting a list of type t1 to a list of type t2.

   Your program should typecheck *without* subtyping (i.e., you should use roll and unroll along with case, pair operations, etc.).

   You may use `t1` and `t2` as abbreviations for their definitions if you wish.

   20 points

Name:____

2. Consider the following proposed changes to System F separately and explain why each is a bad idea.

   (a) Replace the typing rule on the left with the typing rule on the right:

   $$\frac{\Delta, \alpha; \Gamma \vdash e : \tau_1}{\Delta; \Gamma \vdash \Lambda\alpha.\ e : \forall\alpha.\tau_1} \qquad\qquad \frac{\Delta; \Gamma \vdash e[\tau_2/\alpha] : \tau_1}{\Delta; \Gamma \vdash \Lambda\alpha.\ e : \forall\alpha.\tau_1}$$

   (b) Replace the typing rule on the left with the typing rule on the right:

   $$\frac{\Delta; \Gamma \vdash e : \forall\alpha.\tau_1 \quad \Delta \vdash \tau_2}{\Delta; \Gamma \vdash e[\tau_2] : \tau_1[\tau_2/\alpha]} \qquad\qquad \frac{\Delta; \Gamma \vdash e : \forall\alpha.\tau_1 \quad \Delta \vdash \tau_2}{\Delta; \Gamma \vdash e[\tau_2] : \forall\alpha.\tau_1}$$

   10 points each

Name:____

3. In this problem, *assume* Caml has `letcc` and `throw` in addition to (and separate from) `try` and `raise`.

   Consider the following programs separately. For each:

   - What does it print?
   - What is the type of `f`?

   Partial credit will require explanation of your answers.

   Part (d) is difficult.

   5 points each

   (a)
```
exception Foo
let f () = (print_string "A"; raise Foo)
let x = try f() with Foo -> f()
```

   (b)
```
exception Foo
let f () = (print_string "A"; Foo)
let x = try f() with Foo -> f()
```

   (c)
```
let f () =
  let rec g i k =
    if i > 0
    then (print_string "A"; g (i-1) k; print_string "B"; 7)
    else throw k 7
  in
  (letcc k. g 3 k)
let x = f()
```

   (d)
```
let f () =
  let k = ref None
  let rec g i =
    if i > 0
    then (print_string "A"; g (i-1); print_string "B"; 7)
    else (letcc k2. ((k := Some k2); 7))
  in
  (g 3;
  match !k with None -> 7 | Some k2 -> (k := None; throw k2 7))
let x = f()
```

Name:____

4. Java interfaces do not allow fields, [1] but suppose they did.

   For each rule below, determine if it is sound or unsound. If it is unsound, give a short (around 10 lines, including class and interface definitions) example program that would typecheck but get stuck at run-time. Do not worry about syntax, making a correct main method, etc.

   Recall that if interface I extends interface J, then $I \leq J$.
   Also recall that a `final` field can be read but not written.

   Assume interface J has a field f of type T.

   (a) If f is non-final, interface I may extend interface J by changing f to have a subtype of T.

   (b) If f is non-final, interface I may extend interface J by changing f to have a supertype of T.

   (c) If f is final, interface I may extend interface J by changing f to have a subtype of T.

   (d) If f is final, interface I may extend interface J by changing f to have a supertype of T.

   20 points total, graded together

[1] Technically, Java allows `public static final` fields, but this problem considers instance fields.

Name:____

5. Suppose we *change the semantics* of Java so that method-lookup uses multimethods instead of static overloading.

   True or false. **Briefly explain your answers.**

   (a) If all methods in program P take 0 arguments (that is, all calls look like $e.m()$), then P definitely behaves the same after the change.

   (b) If all methods in program P take 1 argument (that is, all calls look like $e.m(e')$), then P definitely behaves the same after the change.

   (c) If a program P typechecks without ever using subsumption, then P definitely behaves the same after the change.

   (d) Given an arbitrary program P, it is decidable whether P behaves the same after the change.

   5 points each

Name:____

6. In the simply-typed λ-calculus with records and without subtyping, the following is true by inspection of the typing rules:

   If $\cdot \vdash v : \{l_1 : \tau_1, l_2 : \tau_2\}$, then there exist v_1 and v_2 such that v is $\{l_1 = v_1, l_2 = v_2\}$.

   (a) Explain why the statement above is false in the presence of subtyping. In particular, which subtyping rules make it false?

   6 points

   (b) Revise the claim so that it is true but as strong as possible. That is, complete this sentence, "If $\cdot \vdash v : \{l_1 : \tau_1, l_2 : \tau_2\}$, then v is ..." with a fact that requires the assumed typing derivation. You can state the claim in English but be precise.

   6 points

   (c) Prove your revised claim. (Hints: Use a "helper" lemma about subtyping derivations where the supertype is a record type containing certain fields. You will need induction and a strengthend induction hypothesis to prove the claim in part (b); it turns out the helper lemma does not need induction.)

   8 points

Name:____

This page is intentionally blank.