Name:____

CSE 505, Fall 2003, Midquarter Examination
4 November 2003

Please do not turn the page until everyone is ready.

Rules:

- The exam is open-book, open-note, closed electronics.
- Please stop promptly at 11:50.
- You can rip apart the pages, but please write your name on each page.
- You can turn in other pieces of paper.
- There are six questions (all with subparts), worth equal amounts. The subparts are not necessarily worth equal amounts.

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 roughly in the order we covered the material, not necessarily order of difficulty. Skip around.
- If you have questions, ask.
- Relax. You are here to learn, not beat the mean.

Name:____

1. Consider this syntax for IMP expressions, which *has (integer) division as the only arithmetic operator*:

$$e ::= c \mid x \mid e/e$$

   (a) Give a large-step operational semantics of the form $H\ ;\ e \Downarrow c$ for these expressions. Make sure that if evaluation of e under H would involve dividing by 0, then there is no c for which you can derive $H\ ;\ e \Downarrow c$.
   (Hint: You need 3 inference rules.)
   (Note: You may assume $H(x)$ is defined as in class.)

   (b) Now suppose we add an explicit **error** result. Add inference rules to your previous answer so that $H\ ;\ e \Downarrow v$ where $v ::= c \mid \textbf{error}$. Make sure that if evaluation of e under H would involve dividing by 0, then $H\ ;\ e \Downarrow \textbf{error}$.
   (Hint: You need 3 more inference rules, so 6 total.)

   (c) Does adding the rule $\dfrac{}{H\ ;\ 0/e \Downarrow 0}$ change the semantics you defined for (a) and (b)? Explain.

Name:____

2. Here is our *unchanged* syntax and semantics for IMP statements:

$$s ::= \mathsf{skip} \mid x := e \mid s; s \mid \mathsf{if}\ e\ s\ s \mid \mathsf{while}\ e\ s$$

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

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

(a) Define a judgment of the form $\mathrm{mysize(s) = n}$. Informally, n should be: (the number of skip statements in s) plus (*two times* the number of assignment statements in s). For example, you should be able to derive $\mathrm{mysize}(\mathsf{skip}; \mathrm{x := 0; y := 1}) = 5$.
(Hint: You need 5 inference rules.)

(b) Prove the following: If s *has no while-statements or if-statements* and $H\ ;\ s \rightarrow H'\ ;\ s'$ and $\mathrm{mysize(s) = n}$ and $\mathrm{mysize(s') = n'}$, then $n' < n$.
Note: This theorem is true with if-statements (but not while-statements), but you do not have to show this.

(c) Using part (b), argue *informally* (no proof required) that while-free programs terminate.

Name:____

3. Describe what each of the following O'Caml programs would print:

(a)
```
let f x y = x y in
let z = f print_string "hi" in
f print_string "hi"
```

(b)
```
let f x = (fun y -> print_string x) in
let g = f "hi" in
let x = "mom" in
g "pizza"
```

(c)
```
let rec f n x =
  if n>0
  then (let _ = print_string x in f (n-1) x)
  else ()
in
f 3 "hi"
```

(d)
```
let rec f n x =
  if n>0
  then (let _ = print_string x in f (n-1) x)
  else ()
in
f 3
```

(e)
```
let rec f x = f x in
print_string (f "hi")
```

Name:____

4. Consider a λ-calculus with *pairs* built-in. That is, (v_1, v_2) is a value if v_1 and v_2 are values, $(v_1, v_2).1 \rightarrow v_1$ and $(v_1, v_2).2 \rightarrow v_2$.

   (a) Give an encoding of triples that uses pairs. You should define four terms: a three-argument function (using currying) to build a triple, and functions for returning the first, second, and third part of a triple. (By *encoding*, we mean you may *not* extend the syntax of the language.)

   (b) In the simply-typed λ-calcus with pairs (and types of the form $\tau_1 * \tau_2$), give *two different types* that your function for forming a triple could have. (I.e., if e is your term for building a triple, give two τ such that $\cdot \vdash e : \tau$.)

Name:____

5. Under what assumptions do the following terms type-check in the simply-typed λ-calculus? That is, for the given e, describe all Γ and τ such that $\Gamma \vdash e : \tau$.

   (a) $e = x\ y$

   (b) $e = \lambda x.\ (f\ (f\ x))$

   (c) $e = \lambda x.\ (\lambda y.\ x)$

   (d) $e = \lambda x.\ (x\ (\lambda y.\ x))$

Name:____

6. Recall how we extend the simply-typed λ-calculus with *fix*:

$$\frac{e \to e'}{\textit{fix}\ e \to \textit{fix}\ e'} \qquad \frac{}{\textit{fix}\ \lambda x.\ e \to e[(\textit{fix}\ \lambda x.\ e)/x]} \qquad \frac{\Gamma \vdash e : \tau \to \tau}{\Gamma \vdash \textit{fix}\ e : \tau}$$

Also we recall that this extension is type-safe.

(a) If we add the rule

$$\frac{}{\Gamma \vdash \textit{fix}\ e : \tau}$$

is our language still type-safe? If not, give a program that gets stuck. If so, argue the case of the Preservation Lemma proof for a typing derivation ending with this rule.

(b) If we add the rule

$$\frac{}{\Gamma \vdash \textit{fix}\ \lambda x.\ x : \tau}$$

is our language still type-safe? If not, give a program that gets stuck. If so, argue the case of the Preservation Lemma proof for a typing derivation ending with this rule.

Hint: The Preservation Lemma is: If $\cdot \vdash e : \tau$ and $e \to e'$, then $\cdot \vdash e' : \tau$. We prove it by induction on the derivation of $\cdot \vdash e : \tau$.