Name:____

CSE 505, Fall 2006, Midterm Examination
2 November 2006

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 11:50.**
- You can rip apart the pages, but please write your name on each page.
- There are **100 points** total, distributed **unevenly** among **4** 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, make sure you get to all the problems.
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

For your reference:

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

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

CONST $$\frac{}{H\ ;\ c\ \Downarrow\ c}$$

VAR $$\frac{}{H\ ;\ x\ \Downarrow\ H(x)}$$

ADD $$\frac{H\ ;\ e_1\ \Downarrow\ c_1 \qquad H\ ;\ e_2\ \Downarrow\ c_2}{H\ ;\ e_1+e_2\ \Downarrow\ c_1+c_2}$$

MULT $$\frac{H\ ;\ e_1\ \Downarrow\ c_1 \qquad H\ ;\ e_2\ \Downarrow\ c_2}{H\ ;\ e_1*e_2\ \Downarrow\ c_1*c_2}$$

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

ASSIGN $$\frac{H\ ;\ e\ \Downarrow\ c}{H\ ;\ x := e\ \to\ H, x\mapsto c\ ;\ \mathsf{skip}}$$

SEQ1 $$\frac{}{H\ ;\ \mathsf{skip};s\ \to\ H\ ;\ s}$$

SEQ2 $$\frac{H\ ;\ s_1\ \to\ H'\ ;\ s_1'}{H\ ;\ s_1;s_2\ \to\ H'\ ;\ s_1';s_2}$$

IF1 $$\frac{H\ ;\ e\ \Downarrow\ c \qquad c>0}{H\ ;\ \mathsf{if}\ e\ s_1\ s_2\ \to\ H\ ;\ s_1}$$

IF2 $$\frac{H\ ;\ e\ \Downarrow\ c \qquad c\leq 0}{H\ ;\ \mathsf{if}\ e\ s_1\ s_2\ \to\ H\ ;\ s_2}$$

WHILE $$\frac{}{H\ ;\ \mathsf{while}\ e\ s\ \to\ H\ ;\ \mathsf{if}\ e\ (s;\mathsf{while}\ e\ s)\ \mathsf{skip}}$$

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

$\boxed{e\to e'}$

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

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

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

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

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

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

- If $\cdot\vdash e:\tau$ and $e\to e'$, then $\cdot\vdash e':\tau$.
- If $\cdot\vdash e:\tau$, then e is a value or there exists an e' such that $e\to e'$.
- If $\Gamma,x{:}\tau'\vdash e:\tau$ and $\Gamma\vdash e':\tau'$, then $\Gamma\vdash e[e'/x]:\tau$.

Name:____

1. (IMP with booleans)
   In this problem we extend the IMP expression language with booleans: true, false, negation, and inclusive-or. (Variables hold integers or booleans, but that is not directly relevant to the questions below.) The new syntax forms are:

   $$e \quad ::= \quad \ldots \mid \mathsf{true} \mid \mathsf{false} \mid \neg e \mid e \vee e$$

   The result of evaluating an expression can be an integer (not relevant below), true, or false. That is, we have $H \; ; \; e \; \Downarrow \; v$ where $v ::= c \mid \mathsf{true} \mid \mathsf{false}$.

   Negation and inclusive-or can be "stuck" if a subexpression does not evaluate to a boolean.

   (a) (**10** points) Add rules to our large-step operational semantics to support the new syntax forms. For $e_1 \vee e_2$, use short-circuiting left-to-right evaluation (like `||` in many languages). If your rules all contain explicit uses of false and true, then you should expect to write 7 rules.

   (b) (**12** points) Theorem: If e always evaluates to a boolean, then e and $\neg\neg e$ are equivalent.

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

   (c) (**10** points) Add implication ($e \Rightarrow e$) to the language. Recall "a implies b if a is false or b is true."

   - Give large-step operational semantics rules that support this extension "directly," using short-circuiting left-to-right evaluation. If your rules all contain explicit uses of false and true, then you should expect to write 3 rules.
   - Give 1 rule that works just as well as your 3 rules by treating implication as a derived form. Remember this should be a large-step rule. Use v in this rule.

Name:____

(This page intentionally blank.)

Name:____

2. (**18** points) (IMP with large-step semantics)

We can give IMP statements a large-step semantics with a judgment of the form $H; s \Downarrow H'$. The rules below do so, but there are *errors*. (The rules match neither our informal understanding nor our small-step semantics.) Find **three** errors (two of which are the same conceptual error), explain the problem, why it is a problem, and how to change the rules to solve the problem.

SKIP
$$\frac{}{H; \mathsf{skip} \Downarrow H}$$

ASSIGN
$$\frac{H \; ; \; e \; \Downarrow \; c}{H; x := e \Downarrow H, x \mapsto c}$$

SEQ
$$\frac{H; s_1 \Downarrow H_1 \qquad H; s_2 \Downarrow H_2}{H; (s_1; s_2) \Downarrow H_2}$$

IF1
$$\frac{H \; ; \; e \; \Downarrow \; c \qquad H; s_1 \Downarrow H_1 \qquad H; s_2 \Downarrow H_2 \qquad c > 0}{H; \mathsf{if} \; e \; s_1 \; s_2 \Downarrow H_1}$$

IF2
$$\frac{H \; ; \; e \; \Downarrow \; c \qquad H; s_1 \Downarrow H_1 \qquad H; s_2 \Downarrow H_2 \qquad c \leq 0}{H; \mathsf{if} \; e \; s_1 \; s_2 \Downarrow H_2}$$

WHILE
$$\frac{H; \mathsf{if} \; e \; (s; \mathsf{while} \; e \; s) \; \mathsf{skip} \Downarrow H'}{H; \mathsf{while} \; e \; s \Downarrow H'}$$

Name:____

3. (**15** points) (Caml and functional programming)

   Consider this Caml code, which type-checks and runs correctly.

```
type dumbTree = Empty | Node of dumbTree * dumbTree

let rec s f t =
   match t with
     Empty -> f t
   | Node(x,y) -> f t + s f x + s f y

let c1 t = s (fun x -> 1) t
let c2 t = s (fun x -> match x with Node(l,Empty) -> 1 | _ -> 0) t
```

   (a) What are the types of `s`, `c1`, and `c2`?

   (b) What do `c1` and `c2` compute? (Hint: The answers are straightforward.)

   (c) Rewrite the last two lines of the code so they are shorter and equivalent.

Name:____

4. (Coin-flipping in Lambda-Calculus)

In this problem we take the simply-typed lambda-calculus with conditionals (true, false, $\mathsf{if}\ e_1\ e_2\ e_3$, and the type bool) and add a "coin-flip" expression, flip. This expression is not a value. Our call-by-value left-to-right small-step semantics has two new semantic rules:

$$\frac{}{\mathsf{flip} \to \mathsf{true}} \qquad\qquad \frac{}{\mathsf{flip} \to \mathsf{false}}$$

(a) (**5** points) In lambda-calculus with conditionals, write a (curried) function that returns the exclusive-or of its arguments. Do not use the constant true and use the constant false only once. (This does not require flip.)

(b) (**5** points) Argue that for all e, $(\lambda x.\ e)\ \mathsf{true}$ and $e[\mathsf{true}/x]$ are equivalent under call-by-value.

(c) (**8** points) Argue that depending on e, $(\lambda x.\ e)\ \mathsf{flip}$ and $e[\mathsf{flip}/x]$ may or may not be equivalent under call-by-value.

(d) (**5** points) Give a typing rule for flip.

(e) (**12** points) Assuming we have proofs of progress, preservation, and substitution for lambda-calculus with conditionals, explain how to extend the proofs for programs containing flip. Be clear about the induction hypothesis and what cases you are adding.

Name:____

(This page intentionally blank.)