CIS 500 — Software Foundations
Midterm I

October 12, 2005

Name:	____________________
Student ID:	____________________
Email	____________________
Status	____ registered for the course
	____ not registered: trying to improve a previous grade
	____ not registered: just taking the exam for practice
Program	____ undergrad
	____ undergrad (MSE submatriculant)
	____ CIS MSE
	____ CIS MCIT
	____ CIS PhD
	____ other

	Score
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
Total	

Instructions

- This is a closed-book exam: you may not make use of any books or notes.
- You have 80 minutes to answer all of the questions. The entire exam is worth 80 points.
- Questions vary significantly in difficulty, and the point value of a given question is not always exactly proportional to its difficulty. Do not spend too much time on any one question.
- Partial credit will be given. All correct answers are short. The back side of each page may be used as a scratch pad.
- Good luck!

Operational semantics

The first three questions concern the following simple programming language:

`t` ::=		*terms*
	`true`	*constant true*
	`false`	*constant false*
	`if t then t else t`	*conditional*
	`pair t t`	*pairing*
	`fst t`	*first component*
	`snd t`	*second component*
`v` ::=		*values*
	`true`	*true value*
	`false`	*false value*
	`pair v v`	*pair value*

and its *large-step* operational semantics.

$$\texttt{true} \Downarrow \texttt{true} \tag{B-True}$$

$$\texttt{false} \Downarrow \texttt{false} \tag{B-False}$$

$$\frac{\texttt{t}_1 \Downarrow \texttt{true} \qquad \texttt{t}_2 \Downarrow \texttt{v}}{\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Downarrow \texttt{v}} \tag{B-IfTrue}$$

$$\frac{\texttt{t}_1 \Downarrow \texttt{false} \qquad \texttt{t}_3 \Downarrow \texttt{v}}{\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \Downarrow \texttt{v}} \tag{B-IfFalse}$$

$$\frac{\texttt{t}_1 \Downarrow \texttt{v}_1 \qquad \texttt{t}_2 \Downarrow \texttt{v}_2}{\texttt{pair } \texttt{t}_1 \ \texttt{t}_2 \Downarrow \texttt{pair } \texttt{v}_1 \ \texttt{v}_2} \tag{B-Pair}$$

$$\frac{\texttt{t} \Downarrow \texttt{pair } \texttt{v}_1 \ \texttt{v}_2}{\texttt{fst } \texttt{t} \Downarrow \texttt{v}_1} \tag{B-Fst}$$

$$\frac{\texttt{t} \Downarrow \texttt{pair } \texttt{v}_1 \ \texttt{v}_2}{\texttt{snd } \texttt{t} \Downarrow \texttt{v}_2} \tag{B-Snd}$$

1. (5 points) Show the derivation of the *large-step* evaluation of the following term.

   ```
   fst (if true then pair true false else pair false true)
   ```

2. (10 points) We might also want to define a *small-step* semantics for this language, such that

$$\mathtt{t} \Downarrow \mathtt{v} \text{ if and only if } \mathtt{t} \longrightarrow^{*} \mathtt{v}$$

Recall that a small-step semantics is composed of both computation and congruence rules. The congruence rules for this language are as follows:

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{if}\ \mathtt{t}_1'\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3} \qquad \text{(E-IF)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{pair}\ \mathtt{t}_1\ \mathtt{t}_2 \longrightarrow \mathtt{pair}\ \mathtt{t}_1'\ \mathtt{t}_2} \qquad \text{(E-PAIR1)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{pair}\ \mathtt{v}\ \mathtt{t}_2 \longrightarrow \mathtt{pair}\ \mathtt{v}\ \mathtt{t}_2'} \qquad \text{(E-PAIR2)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{fst}\ \mathtt{t}_1 \longrightarrow \mathtt{fst}\ \mathtt{t}_1'} \qquad \text{(E-FST)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{snd}\ \mathtt{t}_1 \longrightarrow \mathtt{snd}\ \mathtt{t}_1'} \qquad \text{(E-SND)}$$

Some of the computation rules are:

$$\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{t}_2 \qquad \text{(E-IFTRUE)}$$

$$\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{t}_3 \qquad \text{(E-IFFALSE)}$$

However, this list is not complete. What are the remaining computation rules for the small-step semantics?

3. (5 points) With a small step semantics, there is always the possibility that a term could fail to produce a value. Are there any “stuck” terms in this language? If so, give an example. If not, explain why not.

Functional programming

The following questions are about the untyped lambda calculus. For reference, the semantics of this language appears at the end of the exam.

Recall the Church encoding of lists and booleans in the untyped lambda calculus.

```
tru = λx. λy.x
fls = λx. λy.y
not = λb. b fls tru
and = λb1. λb2. b1 b2 fls
or  = λb1. λb2. b1 tru b2

nil = λc. λn. n
cons = λh. λt. λc. λn. c h (t c n)
head = λl. l (λh.λt. h) fls
tail = λl.
          fst (l (λx. λp. pair (snd p) (cons x (snd p)))
                 (pair nil nil))
isnil = λl. l (λh.λt. fls) tru
```

4. (5 points) Which of the following terms defines the function `all` that takes a list of boolean terms and determines of **all** of the terms are true? For example,

   `all (cons tru (cons fls nil))` should be equivalent to `fls`

   and

   `all nil` should be equivalent to `tru`. Circle the correct answer.

   (a) `all = λl. l and tru`

   (b) `all = λl. all (hd l) (tail l)`

   (c) `all = λl. l (λa.λb. a tru b) fls`

   (d) `all = λl. (λa. λb. a and b) l fls`

5. (5 points) Which of the following terms defines the function `map` that take a term `l`, representing a list, and a function `f`, applies `f` to each element of `l`, and yields a list of the results (just like the `List.map` function in OCaml). For example:

   `map not (cons tru (cons fls nil))`

   should be equivalent to

   `(cons fls (cons tru nil))`. Circle the correct answer.

   (a) `map = λf. λl. l (f cons) nil`

   (b) `map = λf. λl. l (λh. λt. cons t (f h)) nil`

   (c) `map = λf. λl. λc. λn. l (λh. λt. c (f h) t) n`

6. (5 points) Which of the following OCaml terms implements the map function, using recursion? Circle the correct answer.

   (a) `let rec map f l = match l with [] → f [] | (hd :: tl) → f hd :: map f tl`
   (b) `let rec map f l = match l with [] → [] | (hd :: tl) → f hd :: map f tl`
   (c) `let rec map f l = match l with [] → f [] | (hd :: tl) → hd :: map f tl`
   (d) `let rec map f l = match l with [] → [] | (hd :: tl) → hd :: map f tl`
   (e) `let rec map f l = match l with [] → f [] | (hd :: tl) → f hd :: f tl`
   (f) None of the above

Proofs by induction

7. (4 points) What is the structural induction principle for the untyped lambda calculus?

8. (15 points) Complete the following proof of a property of the untyped lambda calculus, by induction on the structure of lambda terms.

   Theorem: If $\texttt{t}$ is closed, and $\texttt{t} \longrightarrow \texttt{t}'$, then $\texttt{t}'$ is closed.

   You may use, without proving, the following lemma about substitution.

   Lemma: If $\lambda\texttt{x}.\texttt{t}_1$ is closed, and $\texttt{t}_2$ is closed, then the substitution $[\texttt{x} \mapsto \texttt{t}_2]\texttt{t}_1$ is also closed.

   We prove the theorem by induction on the structure of the lambda term $\texttt{t}$.

   - Suppose $\texttt{t}$ is a variable $\texttt{x}$. This case is trivial because

   - Suppose $\texttt{t}$ is a lambda term $\lambda\texttt{x}.\ \texttt{t}_1$. This case is also trivial because

   - Suppose $\texttt{t}$ is an application $\texttt{t}_1\ \texttt{t}_2$. Consider the possible ways that $\texttt{t} \longrightarrow \texttt{t}'$.
     - Suppose the last rule used was E-App1 where $\texttt{t}_1 \longrightarrow \texttt{t}_1'$.

– Suppose the last rule used was E-App2, where $\mathtt{t}_1$ is a value and $\mathtt{t}_2 \longrightarrow \mathtt{t}_2'$.

– Suppose the last rule used was E-AppAbs, where $\mathtt{t}_1$ is a lambda term $\lambda\mathtt{x.}\ \mathtt{t}_{11}$, $\mathtt{t}_2$ is a value, and $\mathtt{t}'$ is $[\mathtt{x} \mapsto \mathtt{t}_2]\mathtt{t}_{11}$.

Untyped lambda-calculus

9. (9 points) What do the following lambda calculus terms step to, using the *single-step* evaluation relation $\mathtt{t} \longrightarrow \mathtt{t}'$. Write *NONE* if the term does not step. For reference, the semantics of the untyped lambda calculus appears in the appendix of the exam.

   (a) `(λx.x)(λx. x x)(λx. x x)`

   (b) `(λx. (λx.x) (λx. x x))`

   (c) `(λx. (λz. λx. x z) x) (λx. x x)`

10. (9 points) Now consider the leftmost/outermost evaluation relation from homework 4.

$$\frac{\mathtt{t_1} \longrightarrow \mathtt{t_1}'}{\mathtt{t_1\ t_2} \longrightarrow \mathtt{t_1}'\ \mathtt{t_2}}\ \text{E-App1} \qquad \frac{\lambda\mathtt{x.t_1} \not\longrightarrow}{(\lambda\mathtt{x.t_1})\ \mathtt{t_2} \longrightarrow [\mathtt{x} \mapsto \mathtt{t_2}]\mathtt{t_1}}\ \text{E-App2} \qquad \frac{\mathtt{t_2} \longrightarrow \mathtt{t_2}'}{\mathtt{x\ t_2} \longrightarrow \mathtt{x\ t_2}'}\ \text{E-App3}$$

$$\frac{(\mathtt{s\ t}) \not\longrightarrow \qquad \mathtt{t_2} \longrightarrow \mathtt{t_2}'}{(\mathtt{s\ t})\ \mathtt{t_2} \longrightarrow (\mathtt{s\ t})\ \mathtt{t_2}'}\ \text{E-App4} \qquad \frac{\mathtt{t_1} \longrightarrow \mathtt{t_1}'}{\lambda\mathtt{x.t_1} \longrightarrow \lambda\mathtt{x.t_1'}}\ \text{E-Abs}$$

Using this reduction relation, what do the following terms step to? Again, write *NONE* if the term does not step.

   (a) `(λx.x)(λx. x x)(λx. x x)`

   (b) `(λx. (λx.x) (λx. x x))`

   (c) `(λx. (λz. λx. x z) x) (λx. x x)`

Nameless representation of terms

11. (4 points) Suppose we have defined the naming context $\Gamma =$ `a,b,c,d`. Then the "nameless representation" of the term `λx.d x (λy.x)` is `λ.1 0 (λ.1)`.

    Write down the nameless representation for each of the following terms, in the given naming context.

    (a) `λx. λy. x c y`

    (b) `λx. b (λy. d x x) d`

12. (4 points) Write down (in de Bruijn notation) the normal form of the following de Bruijn term:

    `(λ. λ. 1 (λ. 1)) (λ. 0)`

For reference: Untyped lambda calculus

Syntax

t ::=		*terms*
	x	*variable*
	λx.t	*abstraction*
	t t	*application*
v ::=		*values*
	λx.t	*abstraction value*

Evaluation

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1\ \mathtt{t}_2 \longrightarrow \mathtt{t}_1'\ \mathtt{t}_2} \qquad \text{(E-App1)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{v}_1\ \mathtt{t}_2 \longrightarrow \mathtt{v}_1\ \mathtt{t}_2'} \qquad \text{(E-App2)}$$

$$(\lambda \mathtt{x}.\mathtt{t}_{12})\ \mathtt{v}_2 \longrightarrow [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12} \qquad \text{(E-AppAbs)}$$