Programming Languages [Fall 2014]
Test I

NAME: ____

Instructions:

1) This test is 7 pages in length.

2) You have 75 minutes to complete and turn in this test.

3) Short-answer questions include a guideline for how many sentences to write. Respond in complete English sentences.

4) This test is closed books, notes, papers, friends, neighbors, etc.

5) Use the backs of pages in this test packet for scratch work. If you write more than a final answer in the area next to a question, circle your final answer.

6) Write and sign the following: "I pledge my Honor that I have not cheated, and will not cheat, on this test."

____

____

Signed: ____

1. [4 points]
What is a metavariable? [1-2 sentences]

2. [6 points]
What is the Curry-Howard Isomorphism? Describe it in the level of detail discussed in class (e.g., including the role of types). [2-4 sentences]

3. [5 points]
Is Currying syntactic sugar in ML? Explain why or why not. [1-3 sentences]

4. [10 points]
Implement `smult: int * int list list -> int list list`, subject to the constraints of *Assignment I at the undergrad level* (so you can ignore canonical forms).

5. [10 points]
Implement `smult: int -> int list list -> int list list`, subject to the constraints of *Assignment II* (so again, you can ignore canonical forms).

6. [20 points]

In ML, implement `f:int list->int list`, which returns its argument list sorted in ascending order. `f` must not use any recursion or let-environments; instead, `f` should use *foldr* (and no other library functions). [Hint: Begin with "`fun f L = foldr`..."]

7. [10 points]

Show an example of variable capture during substitution. Using your example, what should the correct result of the substitution be (assuming we want to avoid capturing variables)?

8. [5+10+10+10 = 35 points]

For the remainder of this test, let's consider a language O having the following syntax.

types $\tau ::= \text{int} \mid \tau_1 \to \tau_2$ expressions $e ::= n \mid \lambda x{:}\tau.e \mid x \mid e_1\ e_2$

The notation $\lambda x{:}\tau.e$ refers to an *anonymous* function having parameter x (of type τ) and body e. Language O is a simplification of diML as defined in class; whatever features O has are shared with diML (so for example, O is call-by-value).

(a) At what level have we defined O's syntax?

(b) Define O's (SOS-style) dynamic semantics. Use the judgment form $e \to e'$. Assume that capture-avoiding substitution ($[e/x]e'$) is already defined, so you need not define it.

(c) The rules for multi-step evaluation (i.e., zero or more steps) can be defined as follows.

$$\boxed{e \to^* e'} \qquad \frac{}{e \to^* e}\ \text{R} \qquad \frac{e \to^* e'' \quad e'' \to e'}{e \to^* e'}\ \text{T}$$

Prove the following. Lemma 1. $\forall e_1, e_2, e_3{:}\ (e_1 \to e_2 \wedge e_2 \to^* e_3) \Rightarrow (e_1 \to^* e_3)$

Again, the multi-step rules are defined as follows.

$$\boxed{e \rightarrow^* e'} \qquad \frac{}{e \rightarrow^* e}\ \text{R} \qquad \frac{e \rightarrow^* e'' \quad e'' \rightarrow e'}{e \rightarrow^* e'}\ \text{T}$$

(d) Assume Lemma 1 ($\forall e_1, e_2, e_3$: $(e_1 \rightarrow e_2 \wedge e_2 \rightarrow^* e_3) \Rightarrow (e_1 \rightarrow^* e_3)$) holds. Now prove Lemma 2. $\forall e_1, e_2, e_3$: $(e_1 \rightarrow^* e_2 \wedge e_2 \rightarrow^* e_3) \Rightarrow (e_1 \rightarrow^* e_3)$

[Undergraduates stop here. The following problems are for graduate students.]

(e) An alternative way to define dynamic semantics is with a *big-step* relation, having the judgment form $e \Downarrow v$, which means that expression e evaluates to value v. For example, `5`$\Downarrow$`5` and `(λx:int.x)((λx:int.x)5)`$\Downarrow$`5`. Define the $e \Downarrow v$ judgment for O. [8 points] [Hint: We'll never evaluate lone variables, so $x \Downarrow v$ should never be derivable.]

(f) Assume the following lemmas hold.

Lemma 1. $\forall e_1, e_2, e_3$: $(e_1 \to e_2 \wedge e_2 \to^* e_3) \Rightarrow (e_1 \to^* e_3)$

Lemma 2. $\forall e_1, e_2, e_3$: $(e_1 \to^* e_2 \wedge e_2 \to^* e_3) \Rightarrow (e_1 \to^* e_3)$

Lemma 3. $\forall e_1, e_1', e_2$: $(e_1 \to^* e_1') \Rightarrow (e_1\ e_2 \to^* e_1'\ e_2)$

Lemma 4. $\forall x, \tau, e, e_2, e_2'$: $(e_2 \to^* e_2') \Rightarrow ((\lambda x{:}\tau.e)\ e_2 \to^* (\lambda x{:}\tau.e)\ e_2')$

Now prove that $\forall e, v$: $(e \Downarrow v) \Rightarrow (e \to^* v)$. [12 points]