**Programming Languages [Fall 2012]**

**Test III**

**NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructions:**

1) This test is 10 pages in length.

2) You have 2 hours 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, phones, 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. Essay (graded on accuracy, thoroughness, and readability) [20 points]

Discuss Turing completeness, hitting all the main points we discussed throughout the semester.

2. [25 points]

Let L be the version of diML having the following sorts of expressions.

e ::= true | false | if e1 then e2 else e3 | n | e1+e2 | e1<e2 | fun f(x:τ1):τ2=e | e1 e2 | x | ( ) |

(e1,e2) | let(x1,x2)=e1 in e2 end | (inl e):τ1+τ2 | (inr e):τ1+τ2 | case e of (inl x1=>e1|inr x2=>e2) |

roll(e) | unroll(e) | l | ref(e) | !e | e1:=e2 | (e1; e2; …; en) | while(e1){e2} | fail | try e1 with e2 |

return e | {e} | arr[e1].init[e2] | e1[e2] | l n

(a) Formally state L’s type-safety property.

(b) Implement in L a function F that takes an array A of ints and returns the last element of A.

(c) Define all the parts of L specifically related to recursive types.

(d) Define all the parts of L specifically related to references.

3. [15 points]

Scott numerals are an alternative to Church numerals. They can be defined as:

0 ≡ λx.λy.(x) 1 ≡ λx.λy.(y 0) 2 ≡ λx.λy.(y 1) etc.

(a) Define a λUT function Z that takes a Scott numeral N and returns a Church boolean indicating whether N is 0.

(b) Using the normal-order strategy, trace the evaluation of Z 2. As always, show each step of the evaluation, underline redexes, and define and use abbreviations when convenient.

(c) Define a λUT function S that takes a Scott numeral and returns its successor.

(d) Using the normal-order strategy, trace the evaluation of S 2.

(e) Define a λUT function P that takes a Scott numeral and returns its predecessor (or 0 if the argument is 0).

(f) Using the normal-order strategy, trace the evaluation of P 2.

4. [15 points]

A subtyping relation is a binary, reflexive, and transitive relation over types (in other words, it is a *preorder* over types). We write τ1 τ2 when τ1 is a subtype of τ2, which intuitively means that values of type τ1 can safely “stand in for” values of type τ2. For example, if a language has the types *int* and *real*, we could define *int* *real* to mean that integer values can always be used in places where reals are expected (i.e., *int*-type expressions would also have type *real*). Subtyping is used in object-oriented PLs; if A is a subclass of B then the A type is a subtype of the B type.

Now suppose a language called FIBR has types for functions, *int*s, binary products, and *real*s. Below is a definition of when one type is a subtype of another in FIBR, except that the last two rules are missing one or more premises. Complete the rules by adding the missing premises.

τ1 τ2

*int* *real* *int* *int real* *real*

τ1×τ2 τ3×τ4

τ1→τ2 τ3→τ4

5. [25 points]

State and prove progress and preservation for the CBN λST having types for *int*s and functions. You may assume without proof all the standard lemmas. If you run out of space on this page, continue your proofs on the next page.

(This page is intentionally left blank, to provide additional space for Problem 5.)

**[Undergraduates stop here. The following problem is for graduate students.]**

6. [20 points]

Prove or disprove the following statement about the CBV λST having types for *int*s and functions:  
For all Γ and e, there exists at most one τ such that Γ├ e : τ.