Programming Languages [Fall 2016]
Test III

NAME: ____

Instructions:

1) This test is 8 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, smartwatches, 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. [6 points]
How did we define “programming language” in class? Based on that definition, how would one define a particular programming language—what exactly needs to be defined? [2-4 sentences]

2. [5 points]
What are all the ways we discussed for making λ_{ST} Turing complete? [1-2 sentences]

3. [9 points]
For each of the following ML functions, write the function’s type; if the function is ill typed, write “no type”.
a) fun f w x y z = z (x y) (z (y x) (w z))

b) fun f w x y z = (x z) (y (y w) (y z)) z

c) fun f w x y z = (y x z) (w x z) (w z x) w z x

4. [10 points]
Based on the following function names and types, and our in-class discussions, you should be able to deduce the functions' intended behaviors. Implement each function as a single line of ML code that invokes a fold function. As on Assignment II, your implementations may not be recursive; all recursion must occur within folds. Avoid the @ operator.

(a) `mapPartial : ('a -> 'b option) -> 'a list -> 'b list`

(b) `disjointUnion : ('a -> bool) -> 'a list -> ('a list * 'a list)`

5. [30 points]
For the remainder of this test, you can assume that all uses of N refer to natural numbers (either zero or the successor of a natural number), so none of your inference rules need to contain explicit judgments of the form N nat.

(a) Define deductive systems for subtraction and ≤. The judgment forms are N_1-N_2=N_3 and $N_1 \leq N_2$. Subtracting a larger number from a smaller number should be undefined; i.e., N_1-N_2=N_3 should be underivable if N_2 is larger than N_1 (regardless of what N_3 is). The rules defining one judgment form may not use any other judgment forms, so for example, the rules for subtraction may not use ≤ judgments. Also, avoid using more rules than are necessary.

(b) Using your deductive systems, prove that if N_1-N_2=N_3 then $N_3 \leq N_1$.

6. [40 points]
Consider a CBV λ_{ST} having *unit* as the base type, but then with four additional features added: return expressions, pointers, exceptions, and (iso-)recursive types. Define first-order abstract syntax and static and dynamic semantics for this language, as would be appropriate for a proof of type safety. The following page contains additional space for your response.

[This page provides extra space for Problem 6.]

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

7. [20 points]
State all the lemmas/theorems/corollary for proving type safety for the language from Problem 6. For every lemma/theorem/corollary T stated, provide the technique you'd use to prove T. The following page provides extra space for your response, if needed.

[This page provides extra space for Problem 7.]