Programming Languages [Fall 2016]
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. [5 points]
What is a programming language? [1-2 sentences]

2. [5 points]
How does dynamic scoping work? [1-2 sentences]

3. [6 points]
Show how to rewrite any expression of the form (e_1; e_2; e_3; e_4) in ML, using only ML's syntax for defining and invoking anonymous functions.

4. [26 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 x y z = if true then (fn x=>if x<1 then nil else nil) else (fn x=>[true])

b) fun f x y z = x y z

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

d) fun f x y z = y (y x) z

e) fun f x y z = z y y x x

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

5. [15 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 solutions need to contain explicit judgments of the form N nat.

a) Define inference rules for multiplication. The judgment form is $N_1 x N_2 = N_3$. Assume that rules for addition, $N_1 + N_2 = N_3$, have already been defined.

b) Define inference rules for exponentiation, $N_1 \hat{} N_2 = N_3$. Hint: 0^0=1 and 0^1=0.

6. [30 points]
We want to prove this theorem: If $(N_1+N_2 \geq N_1+N_3)$ then $(N_2 \geq N_3)$, for all N_1,N_2,N_3.
Define deductive systems for concluding $N_1+N_2=N_3$ and $N_1 \geq N_2$; then prove the theorem.
Hint: The theorem stated above uses different syntax than the judgment forms, so you must restate the theorem in a way that only uses the syntax of the judgment forms.

7. [13 points]
Consider the following ML function.

```
fun f L =
  let fun g L R [] _ = (L,R)
        | g L (r::rs) (x::xs) true = g (L@[r]) (rs@[x]) xs false
        | g L R (x::xs) toggle = g L (R@[x]) xs (not toggle)
  in g [] [] L false
  end;
```

(a) To what does `f[1,2,3,4,5,6]` evaluate?

(b) Rewrite `f` to shift all recursion into a single fold function. Your new version of `f` should obey the constraints of Assignment II (including using no explicit recursion, and not defining any let-environments).

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

(c) [3 points]
Assume that L_1@L_2 runs in time linear in the size of L_1. Using big-O notation, what's the running time of `f`?

(d) [7 points]
Rewrite `f` to be asymptotically faster, subject to the constraints of Assignment I (including not using library funs like *foldr*, *foldl*, etc., but where let-environments and recursion are allowed).