Comprehensive Exam: Programming Languages Autumn 2004

This is a 30-minute closed-book exam and the point total for all questions is 30.

All of the intended answers may be written within the space provided. (*Do not use a separate blue book.*) Succinct answers that do not include irrelevant observations are preferred. You may use the back of the preceding page for scratch work. If you to use the back side of a page to write part of your answer, be sure to mark your answer clearly.

The following is a statement of the Stanford University Honor Code:

A. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*

   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

C. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By writing my "magic number" below, I certify that I acknowledge and accept the Honor Code.

____

(Number)

Prob	# 1	# 2	# 3	# 4	Total
Score					
Max	3	4	11	12	30

1. (3 points) *Type inference*

   What is the ML type of this function?

```
fun length nil = 0
|   length (first::rest) = 1 + length rest
```

2. (4 points) *Scoping*

   What is the difference between static and dynamic scope? Name a language or language feature that that uses static scope and one that uses dynamic scope.

3. (11 points) *Typing and Garbage Collection*

   Andrew Appel proposed an interesting scheme for representing the run-time type information used in garbage collection. There are two parts to his approach. First, the compiler generates tables that contain the types of global and stack allocated variables. Second, the compiler generates a compact representation of all the types in the form of a type graph. This graph contains the parse trees (or parse graphs) of all the types used in the program. The idea is that together, these two pieces of information should be sufficient to find the type of a data item at run time.

   For example, consider this simple program with one type and one stack allocated variable

```
TYPE t = RECORD
                val: INTEGER;
                next: REF t;
              END;
VAR v: REF t; (* v is a stack allocated variable of type REF t *)
```

   where `REF` indicates a pointer. At garbage collection time, the collector looks at all the global variables and stack allocated variables. In this example, that's just the variable `v`. Since the collector knows the types of all stack allocated variables, it knows that `v` points to an object of type `t`. Since `v` is on the stack, the collector knows that the object `v` points to is reachable. Using the type table, the collector can determine that the data pointed to by `v` has type `t`. The type graph tells the collector that this object contains a `next` field of type `REF t`, so the collector can follow the `next` field and mark the next object as reachable.

(a) (3 points) What problem is this solving? Is the goal here to improve the running time of programs, reduce the run-time space requirements or programs, provide additional programming flexibility, or accomplish something else?

(b) (2 points) Would this work better for Lisp, Scheme, or ML? (Don't worry about polymorphic functions in ML.) Why?

(c) (3 points) Could you use this approach for garbage collecting C programs? If so, how is way of representing type information and using it at collection time helpful (in comparison with the way Scheme or Lisp garbage collectors usually work)? What problem does it not solve?

(d) (3 points) Do you think this will work well for ML program that use polymorphism? Explain what problem might arise with polymorphism.

4. (12 points) *Method Lookup.*
Java and C++ use different implementations of method lookup for objects.

   (a) (4 points) Suppose a Java program constains a declaration `XClass x = new ...` and invokes `XClass` method `m` on object `x`. Explain how the code for `m` is found at run time.

   (b) (4 points) If a C++ program calls virtual member function `f` of object `x`, how is the code for `f` located? Describe the main data structure and briefly explain how it is used.

   (c) (4 points) Describe the trade-off's between the two approaches. Which is more flexible? Which is more efficient? Why is each one appropriate for the language in which it is used.