Comprehensive Exam: Programming Languages Autumn 2002

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. 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	# 3	Total
Score					
Max	7	5	6	12	30

1. (7 points) *Type Inference*

   (a) (3 points) This part of the question asks you to write the ML type of an ML function. Recall that an ML function that takes two integer arguments and returns a Boolean value has type (`int` $*$ `int`) $\rightarrow$ `bool`. What is the ML type for the following function?

   ```
   fun applytwice(f, x) = f(f(x));
   ```

   (b) (2 points) Describe types `<type1>`, `<type2>` and `<type3>` that will make the analogous C code below statically typed. You may write the types using correct or mostly correct C syntax. In particular you can write `<type2> f` in a form where `f` appears in the middle of the type, if you prefer (and if this is correct).

   ```
   <type1> applytwice(<type2> f, <type3> x) {
             return f(f(x));
   };
   ```

   (c) (2 points) Why is the ML type more useful than the C type for `applytwice`? Specifically, what flexibility does the ML type system give, for functions with the kind of type you wrote in (a), that the C type system does not give you for the function in (b)?

2. (5 points) *ALGOL 60 Pass-By-Name*

   In pass-by-name, the result of a procedure call is the same as if the formal parameters were substituted into the body of the procedure. This rule for defining the result of a procedure call by copying the procedure and substituting for the formal parameters is called the *Algol 60 copy rule.* While the copy rule works well for pure functional programs, as illustrated by β-reduction in lambda calculus, the interaction with side effects to the formal parameter may cause unexpected results.

The following Algol 60 code declares a procedure P with one pass-by-name integer parameter. The line `integer x` does not declare local variables – this is just Algol 60 syntax declaring the type of the procedure parameter.

```
begin
   integer i;
   integer array A[1:2];

   procedure P(x);
      integer x;
      begin
         i := x;
         x := i
      end

   i := 1;
   A[1] := 2; A[2] := 3;
   P (A[i]);
   print (i, A[1], A[2])
end
```

(a) (2 points) Explain how the procedure call `P(A[i])` changes the values of `i` and `A` by writing the lines of Algol that you get by substituting the actual parameter for the formal parameter in `P(x)`.

(b) (3 points) What integer values are printed by the program using pass-by-name parameter passing?

3. (6 points) *Control Flow and Memory Management.* An *exception* is a command that aborts part of a computation and transfers control to a handler that was established at some earlier point in the computation.

   (a) (3 points) Why is it easier to write programs with exceptions in a language that provides garbage collection than a language where all data on the heap is explicitly manipulated by the program?

   (b) (3 points) In a concurrent programming language, we have two options:
   (i) raising an exception only affects the current thread, or
   (ii) raising an exception aborts all threads that were spawned below the control point associated with the exception handler.

   Which option would be easier to implement? Which would be more convenient to use? Explain briefly. (This question is only worth 3 points, so do not write more than 3 or 4 sentences.)

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

   (a) (4 points) If a Smalltalk program sends message `m` to object `x`, how is the code implementing `m` located? Describe the main data structure and briefly explain how it is used.

   (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? Your answer should be 3–5 sentences.