Honors Programming Languages  
G22.3110 Fall 2003  

Final Exam

1. (a) Define the terms **static scoping** and **dynamic scoping** and give a very simple example of each.
   
   (b) In Ada (and C++ and Java), exceptions are said to be statically scoped. However, exceptions do traverse the stack (in reverse order of function calls) looking for a handler. In what way, then, are Ada exceptions not **dynamically scoped**? Write an example, in any language with statically scoped exceptions, which illustrates this point.
   
   (c) Suppose your Scheme programming assignment had been to implement a **dynamically scoped** version of Scheme. What would have been different, precisely, in your interpreter?

2. (a) What is the difference between a generic and a non-generic type variable, in the implementation of type inference for ML?
   
   (b) What rule(s) in the ML type system (stated either formally or informally) leads to the requirement of using non-generic type variables?
   
   (c) Answer this based on the type theory lectures at the end of the semester: Write a (simple!) term in the impredicative typed lambda calculus that is not a valid term in the ML type system, and then show that the term is valid in the impredicative calculus.

3. Write, in SETL, a function `subtuple(T)` that, given a tuple `T`, computes the set of all sub-tuples of `T`, where a sub-tuple of `T` contains only elements of `T` and preserves the relative orders of the elements of `T`. For example, `subtuple([1,2,3])` should return
   
   `{[], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]}`.

4. (a) Object oriented programmers often have a hard time understanding the subset interpretation of subtyping, since in C++ and Java, subtypes have extra fields. How would you justify the subset interpretation of subtyping in C++ or Java to a C++ or Java programmer?
   
   (b) In the simply-typed lambda calculus with subtyping, function subtyping is contravariant in the parameter type. Give the type rule that expresses this property of subtyping, and explain function subtyping in terms of the subset interpretation of subtyping.
   
   (c) Given an example that shows that allowing function subtyping to be covariant in the argument type would introduce problems with type safety.

5. (a) In the context of the denotational semantics (using continuations) of a language with assignment, pass by reference, and gotos, explain the difference between a command continuation and an expression continuation.
(b) Suppose that you wanted to add a new syntactic construct, the case statement, to the language (with assignment, pass by reference, and goto) whose denotational semantics we defined in class. Assume the case statement is of the form:

\[
\text{case } e_1: s_1 \mid e_2: s_2 \mid \ldots \mid e_n: s_n \text{ esac}
\]

where the \( e_i \)'s are evaluated in order until some \( e_j \) returns true, at which point \( s_j \) is executed. Give a denotational semantic definition for the case statement (again, within the denotational framework using continuations).

6. Write in Prolog the \texttt{rev} predicate such that

\[
\texttt{rev(L1,L2)}
\]

succeeds if \( L1 \) and \( L2 \) are lists containing the same elements but in reverse order.