Department of Computer Science and Engineering
University of California at San Diego

CSE 230
Winter 1996

Final Examination

WEDNESDAY MARCH 20, 8AM to 11AM.

Your name:

This exam has 5 pages and 50 points in total. Look through the whole exam and answer the questions you find easiest first. Answer each question in the space following it, using the backs of the pages as extra space if necessary.

You may use your own personal lecture notes, the online class lecture notes, your own old assignments, and all handouts and class textbooks. Other books or notes may not be used.

1. (i) [6 points] Comment on the benefits and drawbacks of programming in ML versus in a widely used modern object-oriented language such as C++ or Java.

   In your answer, discuss relevant historical trends and similarities and differences between languages. Use ideas introduced in CSE230 as appropriate. Keep in mind the contrast between programming by a team versus by one person, and the contrast between exploratory hacking and the engineering of long-lasting, robust efficient software.

   (ii) [3 points] Make some reasonable predictions about the design of future programming languages.

2. [Inspired by pp. 302–304 of *Programming Linguistics* by David Gelernter and Suresh Jagannathan, MIT Press, 1990.]

(i) [3 points] Consider the following interactive session with an implementation of Scheme. What can you infer about the semantics of functions in Scheme?

```
==> (define a 2)
2
==> (define b1 (+ a 1))
3
==> (define b2 (lambda () (+ a 1)))
B2
==> (+ b1 (b2))
6
==> (define a 3)
3
==> b1
3
==> (+ b1 (b2))
7
```

(ii) [4 points] Here is another Scheme session:

```
==> (define count 0)
0
==> (define p (delay
                (begin
                   (set! count (+ count 1))
                   (* x 3))))
PROMISE
==> (define x 5)
5
==> count
0
==> (force p)
15
==> count
1
==> (force p)
15
==> count
1
```

According to the Scheme report "`(delay` $\langle expression \rangle$`)` returns an object called a promise which at some point in the future may be asked (by the `force` procedure) to ..." Finish this sentence with a short but precise definition of the semantics of `delay` and `force`.

(iii) [4 points] Show how to implement `delay` and `force` in ML. Note that for this question you will need to use ML updatable variables. You can ignore the dynamic typing of Scheme, and you can assume that all primitive values are integers.

(iv) [2 points] Is it possible to write ML functions named `delay` and `force`, so that the declaration of `p` above could be written

`val p = delay` E `;`

where E is an ML expression? Justify your answer.

3. [4 points] (i) Give the signature of an ML module that is a polymorphic abstract type of association lists. The abstract type should be functional, i.e. non-imperative, and should supply `lookup` and `update` functions.

   (ii) [4 points] Give an ML structure that implements the abstract type for association lists.

4. [Inspired by p. 128 of *ML Primer* by Ryan Stansifer, Prentice-Hall, 1992.]

   The following ML type captures the abstract syntax of expressions in the language SimpleML.

```
type Variable = string;
datatype Expr =
    Var of Variable
  | Num of int
  | Bln of bool
  | Fun of Variable * Expr
  | App of Expr * Expr
  | Cond of Expr * Expr * Expr;
```

   A value of the following ML type represents a type in the SimpleML language.

```
datatype Type =
    Int
  | Bool
  | Typevar of Variable
  | Arrow of Type * Type
```

   (i) [2 points] A type assignment for a SimpleML program (i.e. a SimpleML expression) is a mapping from SimpleML variables to SimpleML types. Show how to instantiate your module for association lists from Question 2 above to give an ML type named `Assignment` whose values represent SimpleML type assignments.

   (ii) [2 points] In the present context, a substitution is a mapping from SimpleML type variables to SimpleML types. Show how to instantiate your module for association lists to give an ML type whose values represent substitutions, named `Substitution`.

   (iii) [8 points] Implement in ML a function that given a SimpleML expression, infers its most general type. To allow for recursive calls, this function should have signature

```
type_of : Substitution * Assignment * Expr -> Substitution * Type
```

   You will need a unification function with signature

```
unify : Substitution * Type * Type -> Substitution
```

   The output of `unify` is an extension of its input substitution such that when this output substitution is applied to the two input SimpleML types, these are equal. If the two input types cannot be unified, `unify` raises an exception. You need not implement this function.

   (iv) [8 points] Note: you may choose to do this question before or after the questions above, depending on which question you think will help you more with the other.

   In the style of denotational semantics, give domain equations and semantic equations that specify the most general type of a SimpleML expression.