May 14, 1990 **Name:** ____________________________

CS 541 — Programming Languages 1

Final Exam

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Answers to essay questions will be judged on their completeness and clarity. However, please do not feel obligated to fill up all the empty space on the page. Conversely, do not feel limited by the space on a page, since you may continue an answer on the back of a page if you wish.

problem	subject	points	score
1	lazy evaluation and streams	15	
2	referential transparency	15	
3	inheritance	15	
4	language design (object-oriented)	35	
5	confluence	10	
6	type errors	10	
Total		100	

This exam (only) counts for 27.5% of your final grade.

1. (15 points) Describe the uses and advantages of lazy evaluation and streams in a functional programming language. Give examples (i.e., describe problems and algorithms for their solution) to illustrate your points.

2. (15 points) Define referential transparency and discuss its significance for the functional programming style. Give examples where appropriate, including examples that illustrate how referential transparency can be violated in an imperative language.

3. (15 points) What are the advantages and disadvantages of an inheritance mechanism as found in an object-oriented programming language.

4. (35 points) In this problem you are to add to the functional subset of Scheme (see below) a primitive `defclass` to define classes as in object-oriented programming (ignore inheritance and class methods for this problem), a primitive `make-instance` that can make an instance of a class (i.e., an object), and a primitive `send` that allows a message to be sent to an object using dynamic (late) binding.

   Your tasks are to give the syntax of class definitions, the `make-instance`, and `send` expressions (by a BNF grammar), and explain their semantics using translations into the functional subset of Scheme. Be sure to both explain your semantics in English as well as giving a translation; in particular note how classes are instantiated and how objects are represented.

   The functional subset of Scheme omits from the full language mutation (operations with a `!` in their name, and assignment (i.e., `set!`). You should use the following as the rough skeleton for your definition of the semantic function that implements `send`

```
(define (send object message . args)
  ; bind method to the procedure that implements message in object
  (apply method args))
```

   The dot (`.`) indicates that the argument `args` is to be bound to a list that contains the remaining arguments (in order). The form `(apply method args)` then calls the procedure denoted by `method`, passing it the arguments in `args`.

5. (10 points) What is a confluent (Church-Rosser) system of rewrite rules? Describe the importance of confluent systems of rules for practical implementations of declarative programming languages such as OBJ.

6. (10 points) What is a type error?