Spring, 1994 Name: ______________________

Com S 541 — Programming Languages 1

Test on SML and Functional Programming

Special Directions for this Test

This test has 5 questions and pages numbered 1 through 5.
Before you begin, please take a moment to look over the entire test so that you can budget your time.
For this test you can use anything that is built-in to SML or listed on the inside of the back cover of Paulson's book without writing it down. **You may use any other function in Paulson's book, or from your notes, but to do so you must write it into your answer.**

General Directions from the "Course Specification"

Short answer questions will be graded on the basis of completeness (did you list all the relevant parts of the answer?) soundness (did you make any errors of fact or contradict yourself?) and clarity. You can help ensure clarity by giving examples where possible.
For essays, critical justifications, and the like there may be no one right answer; so the criteria for judgement will be whether your essay is:

- clear (you use examples where appropriate, you write clear sentences, diagrams, and you avoid excess verbosity),
- sound (you start from facts or reasonable assumptions, your logic is convincing), and
- complete (you consider relevant aspects of the problem and alternatives).

The highest grade will be earned by an answer that is, in addition, concise and elegant; that is, answers that say the most with the least amount of excess. Extra consideration will be given to answers that are especially creative. For programming problems, there is usually more than one right answer, hence it is important to present with your program evidence that it is correct. Answers to programming questions will be scored on the following points:

- elegance and clarity (conciseness, using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution, and how clear and convincing your argument is that the code is correct)
- correctness (whether your program solves the given problem correctly in the given programming language)

You should try to eliminate all syntax errors from your programs, as they will affect both the correctness and clarity of your solution; you may receive no points for programs with major syntax errors. Run-time space and time efficiency of programs is not important, unless the problem says so. Also, checking of external inputs, prompting, labeling output, and other conventions for dealing with humans are not important (for this class), unless the problem says so.
The breakdown of points between these categories is roughly as follows.

- 50% for elegance and clarity (using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution),
- 50% for correctness (whether your program solves the given problem correctly in the given subset of the given language)

You are encouraged to use auxiliary subroutines whenever they are helpful.

1. (10 points) Give (a) an example of a useful higher-order function, along with the SML code that implements that function, and (b) explain why the function you chose is useful by explaining what programming in SML would be like without using that function.

2. (15 points) In your own words, briefly explain (a) what referential transparency is, and (b) how it makes program verification easier (than it would be without it).

3. (10 points) What is the result of the following SML expression?

```
let
   datatype 'a seq = Nil | Cons of 'a * (unit -> 'a seq);
   fun mapq f Nil = Nil
     | mapq f (Cons(x,xf)) = Cons(f x, (fn () => mapq f (xf())));
   fun takeq (0, xf) = []
     | takeq (k, Nil) = []
     | takeq (k, Cons(x,xf)) = x::(takeq (k-1,xf()));
   fun iterates f x = Cons(x, (fn () => iterates f (f x)));
in
   takeq(4, mapq (fn i => i * i) (iterates (fn z => z + 1) 0))
end;
```

4. (15 points) Define a polymorphic, curried function, `only_between`, of type

```
('a*'a -> bool) -> 'a -> 'a -> 'a list -> 'a list
```

that acts as in the following transcript of an SML/NJ session. (Don't ask us what "polymorphic" and "curried" mean; you're supposed to know that.)

```
- only_between (op <) 3 5 [1,2,3,4,5,6,5,4,3,2,1];
val it = [4,4] : int list
- only_between (op <=) 3 5 [1,2,3,4,5,6,5,4,3,2,1];
val it = [3,4,5,5,4,3] : int list
- only_between (op >=) "f" "d" ["a","b","c","d","e","f","g"];
val it = ["d","e","f"] : string list
```

That is, when `only_between` is given a comparison function `comp`, and values `left`, and `right` such that `comp(left,right)` = `true`, it returns a function that can be applied to a list. This function, `(only_between comp left right)`, when applied to a list `lst`, returns a list that is like `lst`, except that all elements of `lst`, x, that are not such that both `comp(left,`x`)` and `comp(`x`,right)`, are removed.

5. (15 points) Consider the following SML datatype declaration.

```
datatype 'a lambdaCalc =
    Var of string
  | Const of 'a
  | Abs of string * ('a lambdaCalc)
  | Appl of ('a lambdaCalc) * ('a lambdaCalc);
```

Define a function `subst_all` of type

```
string -> string -> 'a lambdaCalc -> 'a lambdaCalc
```

such that `(subst_all new old)` is a function that takes an `'a lambdaCalc` and replaces each string equal to `old` in its argument with `new`. (Don't worry about the distinction between free and bound variables.) For example, consider the following transcript of an SML/NJ session.

```
- val xfory = subst_all "x" "y";
val xfory = fn : 'a lambdaCalc -> 'a lambdaCalc
- xfory (Var "y");
val it = Var "x" : 'a lambdaCalc
- xfory (Abs ("y", (Appl((Var "y"), (Var "g")))));
val it = Abs ("x",Appl (Var "x",Var "g")) : 'a lambdaCalc
- xfory (Abs ("y", (Const 1)));
val it = Abs ("x",Const 1) : int lambdaCalc
```