Com Sci 221 — Programming Languages
Final Exam — Winter 1994
Friday, 18 March
2 pages, 6 problems
Open book, open notes, open mind

Each numbered problem is equally valuable.

1. Give three examples, each involving a single procedure declaration and a single call to the procedure that yields different results using the two different parameter-passing methods:

   (a) call by name vs. call by value-result

   (b) call by value-result vs. call by reference

   (c) call by reference vs. call by name

   The body of the procedure should involve only assignments to local variables and formal parameters—no global variables and no input or output. The actual parameters in the procedure call must all represent legitimate l-values, and the call must terminate normally, for both parameter-passing methods. The difference between the parameter-passing methods should lead to different values for one or two variables in the main program immediately after the call to the procedure. The quality of your examples will be judged according to simplicity, brevity, and especially the clarity with which it illustrates the *reasons* for the discrepancies between the parameter-passing methods.

2. (a) Explain briefly but precisely the relationship between postfix notation and machine code for evaluating an expression. Use a simple example to illustrate your explanation.

   (b) Suppose that you are given an *abstract syntax tree* for an arithemtic expression using binary operations. State a simple general principle that determines the number of registers that will be used by machine code to evaluate the expression. The principle should involve easy observations about the shapes of paths in the tree.

3. For each of the following pairs of types, state whether the two are equivalent ($\cong$), the first can represent the second ($\succ$), the second can represent the first ($\prec$), or the two are incomparable ($\perp$). You may use the symbols in parentheses to express your answer, or use English phrases if you prefer. N is the type containing the *nonnegative integers,* $0, 1, 2, \ldots$.

(a)	$(A^*) \times N$	$(A^*) \times (A^*)$
(b)	$(A^*)^*$	$(A + \{0\})^*$
(c)	$N \to A$	A^*
(d)	$A \to (B \to C)$	$(A \times B) \to C$
(e)	$(A \to C) \times (B \to C)$	$(A + B) \to C$
(f)	$A \to (B \times C)$	$(A \to B) \to C$

4. What types will ML derive for the following defined functions?

```
fun compose f g x = f (g x);
fun repeat f = compose f f;
fun threetimes f x = f (f (f x)));
fun notimes f x = x;
val repcomp = repeat (compose repeat);
```

5. Consider the usual iterative program fragment to reverse a list s_0:

$s := s_0$;
$r := empty$;
while $s \neq empty$ **do begin**
    $r := cons(head(s), r)$;
    $s := tail(s)$
end

cons adds a new first element to a list, *head* returns the first element of a list, and *tail* returns all but the first element of a list.

Give appropriate precondition, postcondition, and loop invariant that could be used to show that the program fragment is correct (but don't do the proof). Use the mathematical primitives p^R (reversal of p) and $p \,.\, q$ (p concatenated with q—equivalently q appended to the end of p) to express your assertions.

6. Notice that an infinite list of members of A may be represented by a function from N (nonnegative integers) to A. Show briefly how to implement the lazy stream functions *cons-stream, stream-car,* and *stream-cdr* using the representation of streams as integer functions. You may use Scheme notation, or any other functional notation of your choice.