Com Sci 221 — Programming Languages
Final Exam — Winter 1992
Wednesday, 18 March
2 pages, 7 problems
Open book, open notes, open mind

Each numbered problem is equally valuable.

1. Draw the abstract syntax trees for the expressions, and convert to postfix notation.

   (a) $(a + b) * (c + (d * (e + f)))$

   (b) $(((a + b) * c) + d) * (e + f)$.

   How tall will the stack get when evaluating each of these expressions? State a general principle that determines the height of stack required to evaluate a given postfix expression. Notice that this time you must express stack height in terms of the postfix form, rather than in terms of the abstract syntax tree.

2. 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.

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

3. Consider the following program fragment that computes $\lfloor \log_2 x_0 \rfloor$:

   $x := 1;$
   $y := 0;$
   while $2 * x \leq x_0$ **do begin**
       $x := x * 2;$
       $y := y + 1$
   end

   Give appropriate precondition, postcondition, and loop invariant to show that the program fragment is correct (they are quite simple). Be careful to include *all* essential information in the invariant. *Use your invariant* to explain why the loop condition must be $2 * x \leq x_0$ rather than $x < x_0$.

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

```
fun accumulate f n nil = n
  | accumulate f n (x :: a) = f x (accumulate f n a);
fun append a b = accumulate cons b a;
val flatten = accumulate append nil;
fun compose f g x = f (g x);
val flatten2 = accumulate (compose append flatten) nil;
```

5. What stream is the value of **mystery** in the following Scheme definitions?

```
(define (add-streams s1 s2)
        (cons-stream (+ (head s1) (head s2))
                (add-streams (tail s1) (tail s2))))
(define (integers i)
        (cons-stream i (integers (+ i 1))))
(define mystery
        (cons-stream 0 (add-streams mystery (integers 1))))
```

6. Do problem 4.3 on p. 162 of the text.