Com Sci 221 — Programming Languages
Midterm Exam — Winter 1994
Due Friday, 25 February
3 pages of problems, 5 problems
Open book, open notes, open mind

Take this exam at a time and place of your own choosing, but in time to hand it in at the beginning of class on Friday, 25 February. Give yourself a contiguous 3-hour block of time to do the problems, starting when you open the exam paper. You may refer to your text, notes or any other written materials gathered in advance. You may not communicate with anyone except the instructor about the exam or class material during your 3-hour exam period. Each numbered problem is equally valuable.

1. Draw the abstract syntax tree and postfix forms for the expressions

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

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

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

   How tall will the stack get when evaluating each of these expressions in postfix form? State a general principle that determines the height of stack required to evaluate a given abstract syntax tree when it is translated to postfix.

2. $A \rightarrow B$ denotes the set of functions from A to B (**array** A **of** B), $A \times B$ denotes the set of ordered pairs from A and B (records with two fields, one containing a value in A and the other a value in B), and $A+B$ denotes the marked union of A and B (variant record field containing either a member of A or a member of B). The following 8 types form 4 pairs of essentially equivalent types. What are the pairs?

   (a) $A \rightarrow (B \times C)$

   (b) $(A+B+C) \times (A+B+C)$

   (c) $A \rightarrow (B \rightarrow C)$

   (d) $(A \rightarrow C) \times (B \rightarrow C)$

   (e) $(A \rightarrow B) \times (A \rightarrow C)$

   (f) $\{0,1\} \rightarrow (A+B+C)$

   (g) $(A \times B) \rightarrow C$

   (h) $(A+B) \rightarrow C$

3. The solution to this problem is similar to the clever exponentiation program that we developed in class, but slightly trickier. Be *very* careful with the arithmetic reasoning. It is all simple elementary high-school algebra, but if you make a guess instead of doing the calculation, you are *very* likely to be wrong. All values in this problem are integers.

   Consider a rather peculiar function f mapping two nonnegative integer arguments to a nonnegative integer result, defined recursively by

$$f(v,w) = \begin{cases} 0 & \text{if } v = 0 \text{ or } w = 0 \\ f(v, w/2) * v & \text{if } v, w > 0 \text{ and } w \text{ is even} \\ f(v-1, w-1) + v & \text{if } v, w > 0 \text{ and } w \text{ is odd} \end{cases}$$

   Even though the recursive definition above is *not* a tail recursion, there is a simple iterative way to compute values of f using a single **while** loop with initialization of some variables before the **while** loop. Let v_0 and w_0 be given nonnegative integer values. Use the invariant

$$f(v,w) * x + y = f(v_0, w_0) \text{ and } v \geq 0 \text{ and } w \geq 0$$

   to develop a simple iterative program (a small number of assignments to initialize v, w, x, y, followed by a single **while** loop containing a single **if-then-else** command) computing $f(v_0, w_0)$. You are not required to prove that your program satisfies the invariant. What variable contains the value of $f(v_0, w_0)$ when your loop terminates?

4. Draw the abstract syntax tree and the flowchart for the schematic program fragment

α;
while A **do begin**
    β;
    if B **then begin**
        γ
    end else begin
        δ
    end
end

   Draw a box around the portion of the flowchart that corresponds to each subtree in the abstract syntax tree (that is, to the whole **while** loop, the body of the **while** loop, and the whole **if-then-else** statement—the atomic elements $\alpha, \beta, \gamma, \delta$ are already in boxes). How many flowchart edges go into each box? How many go out of each box?

5. Consider the two types $((A^*)^*)^*$ (lists of lists of lists whose elements are in A) and $(A + \{eol_1, eol_2\})^*$ (lists of elements each of which is either in A, or is one of the special endmarkers eol_1 or eol_2).

   (a) Explain briefly but precisely how every member of $((A^*)^*)^*$ may be represented by a unique member of $(A + \{eol_1, eol_2\})^*$. Don't forget that the empty list ([ ] in ML notation), the list containing the empty list ([[ ]]), and the list containing the list containing the empty list ([[[ ]]]), are three different things.

   (b) Give a brief but precise characterization of the members of $(A + \{eol_1, eol_2\})^*$ that are *not* used to represent members of $((A^*)^*)^*$ in your answer to part a.