Com Sci 221 — Programming Languages
Midterm Exam — Winter 1992
Due Monday, February 17
2 pages of problems, 6 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 Monday, February 17. Give yourself a contiguous 2-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 exam period. Each numbered problem is equally valuable.

1. Draw the abstract syntax tree for the expressions $(a + b) * (c + (d * (e + f)))$ and $(((a + b) * c) + d) * (e + f)$. 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 abstract syntax tree.

2. Suppose that you are designing a language of boolean expressions using the binary infix operators & (*and*) and $\Rightarrow$ (*implies*). Notice that it is very common to express assertions of the form, "A and B and C and D imply E." Also notice that $(A\&B\&C\&D) \Rightarrow E$ is equivalent to $A \Rightarrow (B \Rightarrow (C \Rightarrow (D \Rightarrow E)))$.

   (a) In order to make the form above most convenient to express, while accommodating other forms reasonably as well, fill in the following rules about precedence and associativity:

      i. highest precedence should be assigned to (&, $\Rightarrow$, doesn't matter);

      ii. & should be (left-associative, right-associative, doesn't matter);

      iii. $\Rightarrow$ should be (left-associative, right-associative, doesn't matter).

      In each case, justify your choice in a couple of sentences.

   (b) Suppose now that you must eliminate the &, and use the equivalence mentioned above. Would you change the associativity of $\Rightarrow$? Very briefly, why or why not?

3. Briefly discuss the relations between the set-theoretic types $A + B + (A \times B)$ and $(A + \{nil\}) \times (B + \{nil\})$. Are they equivalent? Nearly equivalent? Does one represent the other?

4. Consider the following program fragment to compute an approximate square root of x (specifically, the largest integer less than or equal to the exact square root of x):

   $y := x;$
   while $y * y > x$ **do begin**
   $\quad y := y - 1;$
   $\quad$ **end**
   end

   Give a pre-assertion and a post-assertion that express the correctness of the program fragment as accurately as possible. Give a loop invariant that can be used to prove that your post-assertion holds for the final state whenever your pre-assertion holds for the initial state of computation. Explain your reasoning briefly. (Hint: it may be easier to express the post-assertion and the invariant using the squaring function, rather than the square-root function.)

5. Consider a sequence of two assignment commands: $expr_1 := expr_2; expr_3 := expr_4$. Under what circumstances does this sequence have exactly the same result when the sequence is reversed ($expr_3 := expr_4; expr_1 := expr_2$)? Consider the possibility that arrays are used on the right and/or left-hand sides of assignments.

6. Give the best example you can of a single procedure declaration and a single call to the procedure that yields 3 different results using call by value-result, call by reference, and call by name. Do not use global variables. The actual parameters in the procedure call must all represent legitimate l-values, and the call must terminate normally, for all three parameter-passing methods. The quality of your example 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.