Com Sci 221 — Programming Languages
Midterm Exam — Winter 1989
Monday, February 13
2 pages, 5 problems
Open book, open notes, open mind

Each numbered problem is equally valuable.

1. The following 6 types form 3 pairs of essentially equivalent types. What are the pairs?

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

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

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

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

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

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

2. Draw the abstract syntax trees for $a+(b+(c+(d+e)))$, and for $(((a+b)+c)+d)+e$. How tall will the stack get when evaluating each of these expressions? Try to state a general principle that determines the height of stack required to evaluate a given abstract syntax tree.

3. Describe, in English or in RAM code, the execution of the assignment command A[i]:=A[A[i]]. Give an example of values for A and i for which A[i]$\neq$A[A[i]] immediately after execution of the command.

4. 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 accomodating 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?

5. Write an iterative (i.e., **while**–loop) program fragment to do binary search on an array of 100 integers that has been sorted into increasing order. The method is outlined on p.

74 of the text, in Example 3.2. Annotate your program fragment with appropriate pre–assertion, post–assertion and loop invariant to express the essentials of its behavior. Hint: try to express all of the assertions in terms of the arithmetic inequalities $\leq$ and/or $\geq$. By doing so, you will simplify their forms, and avoid confusing conditions that depend on whether the element searched for is actually present or not.