Com Sci 221 — Programming Languages
Midterm Exam — Winter 1990
Monday, February 12
1 page, 5 problems
Open book, open notes, open mind

Each numbered problem is equally valuable.

1. For each of the following descriptions of a programming language datatype, give a set-theoretic formula using the basic types **integer** and **character**, any simple enumerations or subranges that you need, and the combining forms $A \times B$ (cross-product), $A + B$ (marked union), $A \rightarrow B$ (functions), A^* (sequences), $\mathcal{P}(A)$ (sets), that best describes the values of that datatype as abstract concepts. Explain briefly (2–3 sentences) any unavoidable difference between the programming language version and the set-theoretic formula (such as values in one that are not allowed in the other, or multiple values in one that represent a single value in the other). Let the primitive datatype **character** contain normal alphabetic, numeric, and other printable characters, but not special items, such as *end-of-line* or *end-of-file*.

   (a) A type of tables with keys each of which is an integer or a name (given as a character string). Associated with each key in the table is a sequence of intermixed integers and names.

   (b) The type **text** representing textual material organized into paragraphs and lines.

   (c) A type of one-dimensional arrays with integer indexes and integer contents, but resizable with operations that change the lower and upper bounds of the range of indexes allowed.

2. Briefly discuss the relation between the set-theoretic types $A+B+C$ and $(A+\{null\})\times(B+\{null\})\times(C+\{null\})$. For instance, is one of them capable of representing the other? How?

3. Draw the abstract syntax tree for the arithmetic expression $2+((3*(5+4))-6)$. Show the steps of evaluation of this expression by rewriting in left-right order. On each tree that occurs in the evaluation, circle the values that would be on the stack just before execution of each arithmetic operation. Double circle the top value on the stack in each case.

4. Draw a simple flowchart that cannot be derived directly from a program involving only assignment, conditional, and **while**-loops. What property of the flowchart makes it impossible that it came from such a structured program?

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.