Programming Language Comprehensive, Feb 1995

INSTRUCTIONS. Choose any 5 of the 6 questions. You have 2.5 hours.

Question 1: Write a `while` program that sets an integer variable x to the number of zeroes in an array segment $b[m]...b[n]$. Don't worry about bounds of the array; we will assume that it contains the segment. You do not need to do a formal proof, but must write the appropriate precondition and postcondition for the program, and state the loop invariant.

Question 2: Define, in a few lines, the concepts of *state* and *environment*. What is the essential difference between these concepts? What kinds of programming language constructs affect the state, and what ones affect the environment?

Question 3: Here is a little **while** program that sets k to x^n, assuming $n \geq 0$.

```
k:=1;
while (n>0) do
  begin
     k:= k*x;
     n:=n-1;
  end
```

The set S of states is the set of all mappings that assign (say, non-negative) integers to the each variable k, n, x. So, for example, if $s \in S$ then $s(x)$ is an integer. This **while** program determines a (partial) function $c : S \rightharpoonup S$ c is the (unique) function such that, for any $s \in S$, $c(s) = s'$ is state where $s'(k) = k^{s(n)}$, $s'(n) = 0$, and $s'(x) = s(x)$.

In defining the semantics of the while loop using approximations, we obtain a number of partial functions $c_i : S \rightharpoonup S$, for i a natural number, such that each c_i is a "finite approximation" of c.

(A). Describe the partial functions c_i for the while loop in this specific program.

(B). What is the sense in which each c_i "approximates" or "is less than" c.

(C). How do all the c_i's, taken collectively, determine c.

Question 4 (A) Describe the three parameter passing strategies known as call-by-name, call-by-value, and call-by-need.

(B) Give some code that behaves differently depending on whether call-by-value or call-by-name is used (this most simply done in LISP or SCHEME notation). This code should not involve assignment or input-output in any form.

(C) Give some code that behaves differently depending on whether call-by-name or call-by-need is used. This code should use assignment, or input-output.

Question 5. In a compiler for an imperative language such as C or Pascal, two kinds of storage allocation mechanisms are typically employed: stack allocation and heap allocation. What are the separate purposes of these? It will be sufficient to name a programming construct or feature that is implemented primarily using stack allocation, and another that is implemented primarily using heap allocation.

Question 6. One of the oldest, simplest, and most effective pasring techniques is known as recursive-descent. In this method, a different procedure is associated with each non-terminal in a grammar. Give a description of how a recursive-descent parser works. In particular, indicate where the "recursion" comes into play. You might want to illustrate with the grammar

$S \rightarrow cAd$
$A \rightarrow ab \mid a$