CS 342 — Principles of Programming Languages March 7, 1990

Test

Please write your name here:

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Answers to essay questions will be judged on their completeness and clarity. However, please do not feel obligated to fill up all the empty space on the page. Conversely, do not feel limited by the space on a page, since you may continue an answer on the back of a page if you wish.

For a multiple choice question, circle the letter of the best answer or answers. Each incorrect answer to a multiple choice question will result in the loss of one point.

problem	subject	points	score
1	regularity	5	
2	structure principle	5	
3	security	6	
4	arrays	6	
5	calling mechanisms	6	
6	scope rules	6	
7	aliasing	6	
8	contour diagrams and calling mechanisms	25	
9	syntax	10	
10	design	25	
Total		100	

This exam (only) counts for 15% of your final grade.

1. (4 points) The pseudo-code language is regular because:
   (a) It is easy to learn, use, and understand.
   (b) Its syntax is described by a regular grammar.
   (c) The ideas used in its design apply with few exceptions.

2. (4 points) What is wrong with the following argument: "the **goto** statement is harmful because it allows one to write programs that are difficult to follow"?
   (a) Nothing, the argument is valid.
   (b) It should say that the **goto** "encourages" programmers to write programs that write programs that are difficult to follow.
   (c) One can write programs that are difficult to follow in any language.

3. (6 points) Which of the following features can cause violations of security? (Circle *all* appropriate choices.)
   (a) Variant records (as in Pascal) that do not have a tag, so the tag can be changed independently of the value.
   (b) Subrange types (as in Pascal) where the values that a variable may take on are constrained to a contiguous subset of some discrete type.
   (c) A primitive that allows one to return storage to the heap (like dispose in Pascal).
   (d) The **goto** statement.
   (e) A type system where the types of arguments to separately compiled subroutines are not checked (as in FORTRAN).
   (f) The call by name mechanism of Algol 60.

4. (6 points) Which of the following are advantages of static arrays (as in FORTRAN and Pascal) as opposed to semidynamic arrays as in Algol 60? (Circle *all* appropriate choices.)
   (a) Space efficiency at run-time.
   (b) Time efficiency at run-time.
   (c) Abstraction.
   (d) Security.
   (e) Orthogonality.

5. (6 points) Which of the following are advantages of call by value-result? (Circle *all* appropriate choices.)

   (a) It is more time efficient at run-time than call by reference for passing large records.

   (b) It is more time efficient at run-time than call by reference for passing integers.

   (c) It is more space efficient at run-time than call by name for passing large arrays.

   (d) It is more space efficient at run-time than call by name for passing integers.

   (e) It is more orthogonal than call by reference.

   (f) It allows one to write strict functions.

   (g) It allows aliasing.

   (h) It prevents aliasing.

6. (6 points) Which of the following are significant advantages static scoping? (Circle *all* appropriate choices.)

   (a) It allows aliasing

   (b) It prevents aliasing

   (c) It makes type checking possible before run-time.

   (d) It delays most type checking until run-time, when it can do the most good.

   (e) It solves the information hiding problems that arise from FORTRAN's COMMON blocks.

   (f) It is more time efficient at run-time than dynamic scoping.

   (g) It prevents electric shocks from carpets that might damage computer memory.

7. (6 points) List three (3) mechanisms (i.e., features) of FORTRAN that can cause aliasing.

8. Consider the following Algol 60 program. Assume that the `printInteger` routine prints its argument on a line of output.

```
begin
    integer i;

    integer procedure ru(b,e);
        boolean b; integer e;
        begin
            ru := e;
            if not b then ru := ru(b,e);
        end;

    integer procedure inc(i);
        integer i;
        begin i := i+1; inc := i
        end;

    i := 1;
    printInteger(ru(i > 3, inc(i)));
end
```

(a) (5 points) Draw the contour diagram for the above program.

(b) (10 points) Describe the behavior of the above program when all parameters are passed by name.

(c) (10 points) Describe the behavior of the above program when the parameters of `ru` are passed by value and the parameters of `inc` are passed by name (assume that actual argument expressions are evaluated left-to-right).

9. Consider the following grammar.

$$\langle\texttt{E}\rangle ::= \langle\texttt{id}\rangle \mid \langle\texttt{combination}\rangle \mid \langle\texttt{abstract}\rangle$$
$$\langle\texttt{combination}\rangle ::= \langle\texttt{E}\rangle\ \langle\texttt{E}\rangle$$
$$\langle\texttt{abstract}\rangle ::= \lambda\ \langle\texttt{id}\rangle\ \texttt{.}\ \ \langle\texttt{E}\rangle$$
$$\langle\texttt{id}\rangle ::= \texttt{x} \mid \texttt{y} \mid \texttt{z}$$

(a) (3 points) Write three (3) different sentences of different lengths in the language generated by the above grammar.

i.

ii.

iii.

(b) (7 points) Prove that the above grammar is ambiguous.

10. (23 points) Consider a design for the type `string`, with the following operations:

- `make_empty()`, which returns the empty string,
- `concat(`c`,`s`)`, which takes a character c and a string s, and returns the string which has c as its first character and the characters of s following.
- `first(`s`)`, which takes a nonempty string s and returns its first character,
- `rest(`s`)`, which takes a nonempty string s and returns a string containing all but the first character of s,
- `length(`s`)`, which returns the number of characters in s, and
- `equal(`s_1`,`s_2`)`, which tests two strings to see if they have the same characters in the same order.

Critically evaluate this design. That is, what are the good points of this design and what are its bad points? What could be done to improve it, if it needs any improvement?