I have neither given nor received unauthorized aid on this examination, nor do I have reason to believe that anyone else has.

_____

Name (printed)

_____

Signature

C S 355
Principles of Programming Languages
Final Exam
Fall 2001

Instructions: This is an closed-book, closed-notes exam.

All work is to be done on these pages. If you need more room for a particular question than is provided, use the backs of these pages but be sure to label your answers clearly. There are a total of 145 points worth of questions on this test. The point values assigned to each question represent a rough indicator of the difficulty and/or level of detail expected. Use your time accordingly.

Your answers, especially where judgments or explanations are requested, should be precise and to the point. Completeness and precision are important, but extraneous material will not be rewarded and may actually cost you points if it gives me the impression that you do not know what part of your own answer is actually important. Remember, it is up to you to demonstrate that you know the material – it is not up to me to try to ferret it out.

1. (10 pts) Throughout this course, we have classified programming languages by paradigms, according to three different properties: Computation Model, Organization, and Parallelism.

   Describe each of the following languages in terms of their computation model and organization:

   (a) C

   (b) Java

   (c) Scheme

   (d) FORTRAN

   (e) SML

   (f) Prolog

2. (5 pts) What are the major phases of translation in a compiler? Which of these are likely to also occur in an interpreter?

3. (9 pts)

   (a) What is "information hiding"?

   (b) What is "encapsulation"?

   (c) What language constructs have been motivated by the desire to support information hiding and encapsulation?

4. (8 pts) Give a regular expression for integer constants according to the following rules:

   - An integer constant may begin with a `+` or `-` sign, or might not have a leading sign.
   - If the first character (after the sign, if there is one) is an 'x', then the number is hexadecimal.
   - If the first character (after the sign, if there is one) is a digit in the range $0-9$, then the number is decimal.
   - Decimal numbers are written with digits $0-9$, and hexadecimal with digits $0-9$ and $A-F$.

   You may use any of the conventional extensions of the basic regular expression notation.

5. (8 pts) Chess games are recorded as a numbered list of pairs of moves (one move by white, followed by one by black). Each move is recorded as a pair of positions, separated by either "-" or "x" (the latter is used to denote a capture). Each position on the 8x8 chessboard is named using a lowercase letter a-h to denote the column and a number 1-8 to indicate the row.
   An example of a game transcript is:

```
1. e2-e4 e7-e5
2. g1-f3 b8-c6
3. f1-b5 d7-d6
4. d2-d4 f8-e6
5. d4xe5 d6xe5
```

   Use basic (not extended) BNF notation to give a grammar for chess game transcripts. The start symbol for your grammar should be `<game>`. Assume that a terminal `integer` has already been defined for positive integers with any number of digits. Your grammar should ignore formatting/layout/whitespace issues.

6. (5 pts) What is meant by the term "reference semantics"? What language that we have studied uses (mainly) reference semantics?

7. (5 pts) At the assembler level, most machines support only unconditional jumps (goto) and conditional jumps (if ... then goto).
   Show how the statement if $\langle exp \rangle$ then $\langle stmt_1 \rangle$ else $\langle stmt_2 \rangle$ would be translated at the assembler level.

8. (8 pts) Type checking may be strong or weak, and may be done statically or dynamically. Answer the questions below with the name of some language that we have studied that closely fits the given description:

   (a) strong, static

   (b) strong, dynamic

   (c) weak, static

   (d) weak, dynamic

9. (12 pts) The following code contains somewhere between 0 and 10 declarations.

```
PROCEDURE P;                    1 2 3 4 5 6 7 8 9 10
VAR
   X: INTEGER;                  1 2 3 4 5 6 7 8 9 10
   Y: INTEGER;                  1 2 3 4 5 6 7 8 9 10

   PROCEDURE Q;                 1 2 3 4 5 6 7 8 9 10
CONST
      Y: INTEGER = Y + 1;       1 2 3 4 5 6 7 8 9 10
   BEGIN
      WRITELN (X + Y);          1 2 3 4 5 6 7 8 9 10
   END;

   PROCEDURE R (X: INTEGER);    1 2 3 4 5 6 7 8 9 10
   BEGIN
      IF X > 1 THEN             1 2 3 4 5 6 7 8 9 10
         R (X-1);               1 2 3 4 5 6 7 8 9 10
      ELSE
         Y := X;                1 2 3 4 5 6 7 8 9 10
      Q;                        1 2 3 4 5 6 7 8 9 10
   END;
BEGIN
   X := 5;                      1 2 3 4 5 6 7 8 9 10
   Y := 4;                      1 2 3 4 5 6 7 8 9 10
   R(Y);                        1 2 3 4 5 6 7 8 9 10
END;
```

Number each of the declarations, marking the number to the left of the declaration. (i.e., to the left of the first declaration, write a "1". To the left of the second, write a "2", etc.).

The numbers 1 through 10 appear to the right of each line of code. For each line, cross out any numbers for which that line is NOT in the scope of that numbered declaration. For example, if the 3rd declaration in the program was for a variable named "`foo`" and line 10 of the program is in the scope of that declaration, then #3 would **not** be crossed out in line 10.

For each line of code, underline the numbers of any declarations whose r-value is retrieved by that line of code. Circle the numbers of any declarations whose l-values (but not r-values) are employed by that line of code. For example, if the 3rd declaration in the program was for a variable named "`foo`" and line 10 of the program uses the rvalue of `foo`, then #3 would be underlined in line 10.

Notes:

- It is possible for some numbers in some lines to have multiple marks.
- Parameters in this program are passed by value.

10. (7 pts) Draw the activation stack with static access links for the program in the previous problem just before the first time that the "`WRITELN`" routine is called. In your picture, show all parameters and local variables.

11. (5 pts) For the same program, draw the activation stack with an accompanying display just before the first time that the "`WRITELN`" routine is called. You may omit parameters and local variables.

12. (8 pts) Give Prolog translations of each of the following statements, all of which describe a set of children's building blocks.

    (a) `a` is a block. So are `b` and `c`.

    (b) `c` is on top of `a`. `a` is on top of `b`.

    (c) One block is over a second block if it is on top of the second one or if it is in top of another block that is over the second one.

    (d) Is there a block that is over `b`?

13. (5 pts) It is often alleged that many C++ programmers are just writing C code and running through a C++ compiler. There are two tell-tale language constructs that one can look for in a programmer's code, such that, if these constructs occur, there is a good chance that the programmer really hasn't embraced the OO style.
    What are these constructs, and what OO language constructs replace them in "proper" C++ code?

14. (8 pts) Consider the following program.

```
program
   I: integer;
   B: array(1..100) of integer;

   procedure Q(X: integer);
   begin
      I := 1;
      X := X + 2;
      B[I] := 5;
      I := 2;
      X := X + 3;
   end;

begin
   B[1] := 1;
   B[2] := 1;
   I := 1;
   Q(B[I]);
   print B[1], B[2];
end.
```

Give each of the outputs for this program that would result if X were passed by

(a) value

(b) reference

(c) value-result

(d) name.

15. (5 pts) Both C and ML boast that their entire syntactic structure for executable code is simply an expression. Note, however, that C provides a construct (namely, a "statement" sequence bracketed by {}) for evaluating one or more expressions in sequence, one right after the other. ML provides no such construct.
    Explain how this difference is indicative of the fundamental nature of these languages.

16. (8 pts) Define the following:
    (a) type expression

    (b) type system

17. (4 pts) What is the essential difference between a module and a class?

18. (5 pts) In C, space on the heap may be allocated using either of two functions:

    malloc(s) is used to allocate space for a single instance of a given type, where s is the number of bytes required to hold an object of that type.

    calloc(s,n) is used to allocate an array of n items, each of size s

    Why is this second function necessary? I.e., why can we not simply call `malloc(s*n)` rather than `calloc(s,n)`?

19. (8 pts) The functors (higher order functions) `map` and `reduce` can be defined in SML as follows:

```
fun map (f, []) = []
  | map (f, a::rest) = f(a)::map(f,rest);

fun reduce (f, [], u) = u
  | reduce (f, x::L, u) = f(x, reduce(f, L, u));
```

    Give a different definition of `map` that uses `reduce` to implement it.

20. (7 pts) Write an Scheme function `(last L)` to return the last element in a list.