Final Exam

9:00 AM – 12:00 Noon, May 1, 1985

Problem 1. (18 points)

For each of the following six programming languages, Ada, BASIC, COBOL, FORTRAN, LOGO, and Pascal, characterize those application areas which are especially well-suited for that language. Mention specific features of the language to justify your claims. (About three sentences per language, please.)

Problem 2. (12 points)

Consider the following Pascal program:

```
program M ;
   var A: array[1..2] of integer ;
   procedure P(Y, Z: integer) ;
      var X: integer ;
      procedure Q(var X: integer) ;
         begin (* body of Q *)
            X := X - A[1] ;
            P(X, X)
         end (* body of Q *);
      begin (* body of P *)
         Z := Y div Z
         Q(A[A[2]]) ;
      end (* body of P *);
   begin (* body of M *)
      A[1] := 2 ;
      A[2] := 1 ;
      P(A[1], A[2])
   end (* body of M *) .
```

If you execute this program, it should (in four or fewer procedure calls) "bomb" when it tries to divide an integer by zero. Draw the stack and display at the moment when the program bombs. Be sure to show on the stack all variables, parameters, and dynamic and static environment chains.

Problem 3. (10 points)

Assume you have a programming language with four operators, ♣, ♢, ♡, and ♠, with left-to-right associativity and the following operator precedence rules:

Highest precedence	♣
Middle precedence	♢ ♡
Lowest precedence	♠

Show both the tree form (Fig. 6-2, page 153) and postfix notation for the expression:

$$(a \diamondsuit b \spadesuit c) \heartsuit d \clubsuit e \diamondsuit (f \clubsuit g)$$

Problem 4. (10 points)

An Ada package has two parts, for example:

```
package INT_STACK_TYPE is
  ...
end INT_STACK_TYPE
package body INT_STACK_TYPE is
  ...
end INT_STACK_TYPE
```

What is the purpose of each of these two parts? Which part(s) must be looked at by (a) someone using the package, and (b) someone modifying the package?

Problem 5. (7 points)

Draw the state of a LOGO interpreter using a reference count garbage collector after it executes each of the following four statements.

```
MAKE "L1 [5 6]
MAKE "L2 (BUTFIRST :L1)
MAKE "L1 (FPUT 4 :L2)
MAKE "L2 (BUTFIRST :L1)
```

Problem 6. (6 points)

Consider the following program written in a generic language:

```
program foo ;
   var n:  integer ;
   function g():  integer;
      begin
         return( n+1 )
      end ;
   function f():  integer;
      var n:  integer ;
      begin
         n := 1 ;
         return( g() )
      end ;
   begin
      n := 5 ;
      print( f() )
   end ;
```

What is the result of executing this program using both (a) dynamic and (b) static scope rules? Explain your answer.

Problem 7. (6 points)

Give at least two good reasons why you can't write a general-purpose quicksort routine in Pascal.

Problem 8. (6 points)

Assume that `x`, `y`, and `z` have been declared to be integer variables in Ada and that `x` and `y` are both positive. Write two or three lines of Ada that will set the integer variable `z` to the minimum of 1000000000 and `x+y`. Use Ada's exception-handling mechanism to make sure that your code works correctly even if the computation of `x+y` signals the Ada exception `OVERFLOW`.

Problem 9. (5 points)

Make a movie showing the stack of temporary values of a compiled Pascal program executing the expression `x+(y*z)-(x mod z)` when `x` is 5, `y` is 6, and `is 7`.

Problem 10. (5 points)

Write a Pascal procedure `increment2` which takes two integer parameters. The effect of `increment2` is to make each actual parameter one greater than it was before the call. For example:

```
increment2(A[i], A[j]) ;
```

is equivalent to:

```
A[i] := A[i] + 1 ;
A[j] := A[j] + 1 ;
```

if `i` and `j` are different and is equivalent to:

```
A[i] := A[i] + 1 ;
```

if `i` and `j` are the same.

Problem 11. (5 points)

Pascal prohibits programmers from directly using the value returned by a pointer-valued function as a pointer reference. For example, you cannot use `f(5)^` in a Pascal expression, even if `f` is function that maps integers into pointers. Can you think of any justification for this restriction? Explain your reasoning. Do you anticipate any problem in implementing an extension of Pascal which does not have this restriction?

Problem 12. (5 points)

It is generally thought that FORTRAN programs are more efficient than Pascal programs; however, it is quite likely that, if you take a Pascal program and translate it into FORTRAN in a straightforward manner (assume the Pascal program had no recursion), the FORTRAN program will use more storage. Discuss briefly (one paragraph) why this is so.

Problem 13. (5 points)

Write FORTRAN `READ` and `FORMAT` statements to input 80 one-digit numbers, typed one per column on a card, into an integer array `C`.