**SECTION A**

1. Describe how to implement a compiler for a language in the same language (“bootstrap- ping”).
2. Invent a syntax for an APL-like matrix-based language that uses ordinary characters.
3. Write a list of interesting operations on strings and compare your list with the predeﬁned operations of SNOBOL and Icon.
4. Write a list of interesting operations on sets and compare your list with the predeﬁned operations of SETL.

**SECTION B**

1. Translate (part of) the BNF syntax of C or Ada into syntax diagrams.
2. Write a program in Pascal or C that compiles and executes, but computes the wrong answer because of a comment that was not closed.
3. Even if Ada used the style of comments used in C and Pascal, bugs caused by not closing comments would be less frequent. Why?
4. In most languages, keywords like begin and while are reserved and may not be used as identiﬁers. Other languages like FORTRAN and PL/I do not have reserved keywords. What are the advantages and disadvantages of reserved words?

**SECTION C**

1. Study your compiler’s documentation and list the optimizations that it performs. Write programs and check the resulting object code for the optimizations.
2. What information does the debugger need from the compiler and linker?
3. Run a proﬁler and study how it works.
4. AdaS is an interpreter for a subset of Ada written in Pascal. It works by compiling the source into P-Code and then executing the P-Code. Study the AdaS program and write a description of the P-Code machine.

**SECTION D**

1. What happens if you execute the following C program on a computer which stores short int values in 8 bits and int values in 16 bits?

 Cshort int i; int j = 280; for (i = 0; i ¡ j; i++) printf(”Hello world”);

1. If a non-standard representation of an enumeration type is used, how would you implement the Ada attribute T’Succ(V)?
2. What will the following program print? Why?

 Cint i = 2; int j = 5; if (i & j) printf(”Hello world”); if (i && j) printf(”Goodbye world”);

1. What is the value of *i* after executing the following statements?

 Cint i = 0; int a[2] = {10,11}; i = a[i++];

**SECTION E**

1. Pascal contains the construct with which opens the scope of a record so that the ﬁeld names can be used directly:

 Pascaltype Rec = record Field1: Integer; Field2: Integer; end; R: Rec;

 with R do Field1 := Field2;(\* OK, direct visibility \*)

 What are the advantages and disadvantages of the construct? Study the Ada renames con- struct and show how some of

 the same functionality can be obtained. Compare the two constructs.

1. Icon has associative arrays called tables, where a string can be used as an array index:

 count[”begin”] = 8; Implement associative arrays in Ada or C.

1. Are the following two types the same?

 Adatype Array Type 1 is array(1..100) of Float; type Array Type 2 is array(1..100) of Float;

 Ada and C++ use name equivalence: every type declaration declares a new type, so two types are declared. Under structural

 equivalence (used in Algol 68), type declarations that look alike deﬁne the same type. What are the advantages and

 disadvantages of the two approaches?

1. An array object of anonymous type (without a named type) can be deﬁned in Ada. In the following example, is the assignment legal? Why?

 AdaA1, A2: array(1..10) of Integer; A1 := A2;

**SECTION F**

1. Simulate a Pascal repeat-statement in Ada and C.
2. The original deﬁnition of FORTRAN speciﬁed that a loop is executed at least one time even if the value of low is greater than the value of high! What could motivate this design?
3. The sequential search in C:

 Cwhile (s[i].data != key) i++;

 might be written as follows:

 Cwhile (s[i++].data != key) ; /\* Null statement \*/

 What is the difference between the two computations?

1. (Scholten) The game of Go is played with stones of two colors, black and white. Suppose that you have a can with an unknown mixture of stones and that you execute the following algorithm:

 Adawhile Stones Left in Can loop Remove Two Stones(S1, S2); if Color(S1) = Color(S2) then Add

 Black Stone; else Add White Stone; end if; end loop;

 Show that the loop terminates by identifying a value which is always decreasing but always non-negative. Can you say

 anything about the color of the last stone to be removed? (Hint: write a loop invariant on the number of white stones.)