Final Exam
10 December, 1990, 6:00-8:30 pm

Open book section
60 Points

Name: _____

This is the open book section of the exam. You may work on it both before and after you turn in the closed book section.

Assume the following set of type definitions and variable declarations for the next two questions:

```
TYPE
     BigArrayType = ARRAY[1..100] OF INTEGER ;
     SmallArrayType = ARRAY[1..20] OF INTEGER ;
     IntPntrType = ^ INTEGER ;
     IntPntrArrayType = ARRAY[1..20] of IntPntrType ;

VAR
     B: BigArrayType ;
     S: SmallArrayType ;
     PB: IntPntrArrayType ;
     PI, PJ: IntPntrType ;
     I, J: INTEGER ;
```

Question I. (8 points)

With the above definitions and declarations, which of the following are valid type-consistent Pascal statements?

```
(a)  J := I ;
(b)  I := PI^ + J ;
(c)  NEW(PB) ;
(d)  PI := PJ ;
(e)  PB[5] := B[5] ;
(f)  PB[5] := PI + 1 ;
(g)  PI := NEW(PJ) ;
(h)  B := S ;
```

Question II. (7 points)

Describe the effect of executing each of the following statements *in sequence*.

```
NEW(PJ) ;
NEW(PI) ;
NEW(PB) ;
PI^ := 5 ;
PJ^ := 6 ;
FOR I := 1 TO 20 DO
     PB[I] := PI ;
PI := PJ ;
I := PI + PJ + PI[17] ;
```

Question III. (15 points)

Write a function called **FirstXX** which takes as its single argument an array of 10 characters, indexed 1 through 10, and returns the index of the first location within the array where the next two elements of the array are the character **'X'**. If there are not two consecutive **'X'**s in the array, your function should return 0.

For example, if the ten elements of the array are:

A B C X X D E X X X

your function should return 4, and if the ten elements are:

A B C D F G H J K L

your function should return 0.

Use the following type definition and header to start your function.

```
TYPE String10Type = ARRAY[1..10] OF CHARACTER ;

FUNCTION FirstXX(A: String10Type): INTEGER ;
```

Now, write a function **HasXX** which takes an array of 10 characters as its single argument and returns **TRUE**, if two consecutive components of that array are **'X'**, and returns **FALSE** otherwise. This time you have to write your own header. You may call **FirstXX** if you find that convenient.

Question IV. (7 points)

Start with the following procedure:

```
PROCEDURE SillyProc(I: INTEGER; VAR J: INTEGER) ;
BEGIN
     I := I+J ;
     J := I*I
END ;
```

Assuming that X and Y Z have been declared as INTEGER variables, describe the effect of executing each of the following statements *in sequence*.

```
     X := 5 ;
     Y := 6 ;
     SillyProc(X, Y) ;
     SillyProc(Y, X) ;
     SillyProc(X, X) ;
```

Question V. (8 points)

The following program will count how many periods appear in the **INPUT** file.

```
PROGRAM CountDot (INPUT, OUTPUT)
VAR
     NumDot: INTEGER ;
     NxtChr: CHAR ;
BEGIN
     NumDot := 0 ;
     WHILE NOT(EOF(INPUT)) DO
          BEGIN
               READ(INPUT, NxtChr) ;
               IF NxtChr = '.' THEN
                    NxtChr := NxtChr + 1
          END ;
     WRITELN(OUTPUT, 'The number of periods is ', NxtChr)
END ;
```

Now, to your task. Modify the above program to count the number of lines that contain a period rather than the number of periods.

Question VI. (15 points)

You are running a successful programming contract business. For each program you write you keep up with three pieces of information:

(1), a fifty character project description,
(2), programming language used, and
(3), lines of code written.

Your immediate goal is the design of a Pascal record to keep up with this information.

First, you use only four programming languages, Pascal, C, COBOL, and BASIC, in your business. Write a definition for a Pascal enumerated type called **ProgLangType** appropriate for the programming language field.

Second, write a definition for a Pascal record type called **JobType** for the storing all three pieces of information.

Finally, you need to send bills for your programming jobs. You decide to change $1 a line for programs written in and Basic, $2 a line for programs written COBOL and C, and $5 a line for programs written in Pascal. All your bills are going to look like the following.

NAME-OF-PROJECT-GOES-HERE
You owe me $*xx.xx* for *N* lines of code.

Now write a few lines of Pascal to generate one of these bills with the information present in a record **Job** of type **JobType**.