4 TOP DOWN DESIGN

Many problems consist of a number of tasks. One good technique in solving such problems is to identify the tasks, decompose each task into sub-tasks and solve these sub-tasks by smaller and simpler solutions. Ultimately, the main tasks and the sub-tasks are converted to program code. In this chapter, we introduce the top down design technique based on problem decomposition and the means to implement such a technique.

4.1 Basic Concepts of Top Down Design

Top down design is a technique that reduces the complexity of large problems. The technique is based on the divide-and-conquer strategy, wherein the problem tasks are divided into sub-tasks repetitively. The division of tasks stops when the sub-tasks are relatively easy to program. The terms *successive refinement or step-wise refinement* also refer to the top-down design technique.

In FORTRAN, each sub-task can be implemented by a separate module. FORTRAN uses two types of program modules, *subroutines* and *functions*. These modules are also called *subprograms*. A typical FORTRAN program consists of a main program with several subprograms. Each subprogram represents a sub-task in the top down design solution.

The top down design process has many advantages:

1. The subprograms can be independently implemented and tested.
2. Subprograms developed by others can be used. For example, a huge library of FORTRAN subprograms known as IMSL (International Mathematical and Statistical Library) is available. The IMSL library has efficient, well tested subprograms for common problems in matrix manipulation, algebraic equations, statistical computations, .. etc.
3. The size of the program is reduced, since identical code segments in the main program are replaced by a single subprogram.

4.2 Subprogram Terminology

There are several new terms with which we should be familiar with while using subprograms. The program file usually consists of a program called the *main program* and all the associated subprograms. These subprograms may appear before or after the main program. A subprogram is *called* or *invoked* by another subprogram or the main program. The calling program passes information to the subprogram through *arguments* or *parameters*. The subprogram returns information to the calling program. In the case of a function, the information which is a single value, is returned as the value of the function name. In the case of a subroutine, the information is returned through some or all the arguments. The arguments that appear in the description of the subprogram are called *dummy* arguments and those that appear in the calling statement are called *actual* arguments. Every subprogram consists of a *header* followed by a *body*. The subprogram body has a statement called the ***RETURN*** statement to return execution control to the calling program. There may be more than one **RETURN** statements in a subprogram. A subprogram ends with an **END** statement.

4.3 Function Subprograms

A function subprogram is the description of a function consisting of several statements. The subprogram computes a single value and stores that value in the function name. A function subprogram consists of a function header and a function body.

4.3.1 Function Header

The *function header* is the first statement of the function and has the following format:

type **FUNCTION** fname (***a list of arguments***)

where

type is the type for the function name (**REAL**, **INTEGER** ..);

fname is the name of the function;

a list of arguments is the optional list of dummy arguments.

If the *type* of the function is not specified, the function type is assumed as either **INTEGER** or **REAL**, as in the case of variables. The rules that apply in naming a variable also apply to function names. If there are no arguments to a function, then the empty parentheses () appear with the function name.

4.3.2 Function Body

The *function body* is similar to a FORTRAN program. It consists of declaration statements, if any, in the beginning, followed by executable statements. Each function body must end with an **END** statement. The **RETURN** statement must appear in the function body at least once. This statement is used to transfer control from the function back to the calling program. The function name should be assigned a value in the function body. A *typical* layout of a function is as follows:

```
      TYPE FUNCTION FNAME (A LIST OF DUMMY ARGUMENTS)
      DECLARATION OF DUMMY ARGUMENTS AND VARIABLES TO BE USED IN THE
      FUNCTION

         EXECUTABLE STATEMENTS
      ..
      ..
      FNAME = EXPRESSION
      ..
      ..
      RETURN
      END
```

4.3.3 *Examples on function subprograms*

Example 1: *Write a real function VOLUME that computes the volume of a sphere ($4/3\pi r^3$) given its radius.*

Solution:

```
      REAL FUNCTION VOLUME(RADIUS)
      REAL RADIUS, PI
      PI = 3.14159
      VOLUME = 4.0 / 3.0 * PI * RADIUS ** 3
      RETURN
      END
```

Example 2: *Write a logical function ORDER that checks whether three different integer numbers are ordered in increasing or decreasing order.*

Solution:

```
      LOGICAL FUNCTION ORDER(X, Y, Z)
      INTEGER X, Y, Z
      LOGICAL INC, DEC
      DEC = X .GT. Y .AND. Y .GT. Z
      INC = X .LT. Y .AND. Y .LT. Z
      ORDER = INC .OR. DEC
      RETURN
      END
```

Example 3: *Write a function subprogram to evaluate the function f(x) defined below.*

$$f(x) = 2x^2 + 4x + 2 \quad \text{if} \quad x < 5$$
$$f(x) = 0 \quad \text{if} \quad x = 5$$
$$f(x) = 3x + 1 \quad \text{if} \quad x > 5$$

Solution:

```
      FUNCTION F(X)
      REAL F, X
      IF (X .LT. 5) THEN
           F = 2 * X ** 2 + 4 * X + 2
      ELSEIF (X .EQ. 5) THEN
           F = 0
      ELSE
           F = 3 * X + 1
      ENDIF
      RETURN
      END
```

4.3.4 Function Call

Let us consider a program consisting of a main program and a function subprogram. The execution of the program begins with the main program. For each call to a function, control is transferred to the function. After the function is executed, the **RETURN** statement ensures that control is transferred back to the calling program. The execution of the main program then resumes at the location the function is called.

Example: *In the following two tables, correct and incorrect function calls to the functions defined in Examples 1, 2 and 3 are given. We assume that in the calling* *program the function names VOLUME, F are declared as* **REAL***, and ORDER as* **LOGICAL***. We also assume A = 5.0, B = 21.0, where A and B are real numbers:*

Examples of correct function calls:

Function Call	**Function Value**
ORDER(3, 2, 4)	.FALSE.
ORDER(3, 4 * 3, 99)	.TRUE.
F(A)	0.0
F(3 + F(2.0))	64.0
VOLUME(B)	38808.0
F(A + B)	79.0

Examples of incorrect function calls:

Incorrect Function Call	**Error Message**
ORDER(3.0, 2, 4)	Argument 1 referenced as real but defined to be integer
F(3.2, 3.4)	More than one argument to function F
VOLUME(5)	Argument 1 referenced as integer but defined to be real

4.3.5 Function Rules

The following rules must be observed in writing programs with function subprograms:

- Actual and dummy arguments must match in type, order and number. The names of these arguments may or may not be the same.
- Actual arguments may be expressions, constants or variable names. Dummy arguments must be variable names and should never be expressions or constants.
- The type of the function name must be the same in both the calling program and the function description.
- The result from the function subprogram, to be returned to the calling program, should be stored in the function name.
- A return statement transfers control back to the calling program. Every function should have at least one return statement.
- The function may be placed either before or after the main program.
- A function is called or invoked as part of an expression.
- A FORTRAN function cannot call itself.

4.3.6 Complete *Examples on function subprograms*

Example 1: *The sum of three integer numbers: Write an integer function SUM to sum three integer numbers. Also write a main program to test the function SUM.*

Solution:

```
C MAIN PROGRAM
      INTEGER X, Y, Z, SUM
      READ*, X, Y, Z
      PRINT*, SUM (X, Y, Z)
      END

C FUNCTION SUBPROGRAM
      INTEGER FUNCTION SUM(A, B, C)
      INTEGER A, B, C
      SUM = A + B + C
      RETURN
      END
```

The execution starts with the reading of variables X, Y and Z in the main program. The execution of the expression SUM(X, Y, Z) transfers control to the function SUM. The value of the actual arguments X, Y and Z is passed to the dummy arguments A, B and C respectively. In the function SUM, execution begins with the first executable statement which computes the value of SUM. The return statement returns control to the main program. The print statement in the main program prints the value of SUM(X, Y, Z) and the execution ends. Assume that the input to the above program is as follows:

```
7 3 9
```

then the output of the program is

```
19
```

Example 2: *Reverse a Two Digit Number: A two digit integer number is to be reversed. A two digit number ranges between 10 and 99. Write a function that first checks if the number is a two digit number and then returns the number with the digits reversed. The function should return an error code -1 if the argument is not a two digit number. Write a main program to test the function.*

Solution:

The main program invokes function RVSNUM after reading a number. If the value returned from the function is -1, an error message is printed. Otherwise, the number and its reversed value are printed. Notice the use of two **RETURN** statements in the function.

```
      INTEGER FUNCTION RVSNUM(NUMBER)
      INTEGER NUMBER, RDIGIT, LDIGIT
      IF (NUMBER .LT. 10 .OR. NUMBER .GT.99) THEN
           RVSNUM = -1
           RETURN
       ENDIF
      LDIGIT = NUMBER / 10
      RDIGIT = NUMBER - LDIGIT / 10 * 10
      RVSNUM = RDIGIT * 10 + LDIGIT
      RETURN
      END

C     MAIN PROGRAM
      INTEGER NUMBER, RVSNUM, RNUM
      READ*, NUMBER
      RNUM = RVSNUM(NUMBER)
      IF (RNUM .EQ. -1) THEN
           PRINT*, 'INPUT ERROR : ', NUMBER
      ELSE
           PRINT*, 'ORIGINAL NUMBER IS ', NUMBER
           PRINT*, 'REVERSED NUMBER IS ', RNUM
      ENDIF
      END
```

If the input to this program is

```
78
```

then the output is:

```
ORIGINAL NUMBER IS 78
REVERSED NUMBER IS 87
```

If the input to this program is

```
123
```

then the output is:

```
INPUT ERROR : 123
```

Note that the actual arguments can be expressions. If the function is invoked with the statement **PRINT***, RVSNUM(4 * 6), the value 42 is printed.

4.4 Special Cases of Functions

There are special cases of functions that do not require subprogram description. These cases may be classified into two groups:

1. Intrinsic (built-in) Functions
2. Statement Functions

4.4.1 Intrinsic Functions

These are predefined functions that are available from the FORTRAN language. Certain functions, such as the trigonometric functions, are frequently encountered in programming. Instead of developing them repeatedly in each program, the language provides these functions. For example, MOD(M,N) is an intrinsic function that requires two integer arguments M and N. The result of the function MOD is an integer value representing the remainder when M is divided by N. A list of commonly used intrinsic functions is given below.

Function	Function Value	Comment
SQRT(X)	Square Root of X	X is a real argument
ABS(X)	Absolute Value of X	
SIN(X)	Sine of angle X	Angle is in radians
COS(X)	Cosine of angle X	Angle is in radians
TAN(X)	Tangent of angle X	Angle is in radians
EXP(X)	e raised to the power X	
LOG(X)	Natural Logarithm of X	X is real
LOG10(X)	Logarithm of X to base 10	X is real
INT(X)	Integer value of X	Converts a real to an integer
REAL(K)	Real value of K	Converts an integer to real
MOD(M, N)	Remainder of M/N	Modulo function

Common Intrinsic Functions

4.4.2 Statement Functions

In engineering and science applications, we frequently encounter functions that can be written in a single statement. For example, $f(x) = x + 2$ is a simple function. In such cases, FORTRAN allows us to write a statement function instead of writing a function subprogram. A *statement function* is defined in the beginning of a program after declaration statements. As a non-executable statement, it should appear before any executable statement. The general form of this statement is as follows:

fname (*a list of arguments*) = expression

where

fname is the name of the function;

a list of arguments is the optional list of dummy arguments; and

expression computes the function value.

The type of the statement function may be declared in the declaration statements. If the type of the function is not declared, it is implicitly defined.

4.4.2.1 Examples of statement functions:

Example 1: *Write a statement function to compute the area of a triangle, given its two sides and an angle.*

```
REAL AREA
AREA(SIDE1,SIDE2,ANGLE) = 0.5 * SIDE1 * SIDE2 * SIN (ANGLE)
```

Example 2: *Write a statement function to compute the total number of seconds, given the time in hours, minutes and seconds.*

Solution:

```
REAL TOTSEC
TOTSEC(HOUR,MINUTE,SECOND) = 3600 * HOUR +60 * MINUTE + SECOND
```

Example 3: *Write a statement function to compute the function* $f(x,y) = 3x^2 + 5xy$

Solution:

```
REAL F
F(X, Y) = 3 * X ** 2 + 5 * X * Y
```

Example 4: *Write a logical statement function to check if three different integer numbers are in increasing or decreasing order.*

Solution:

```
LOGICAL ORDER
ORDER(X,Y,Z) = X.GT.Y .AND. Y .GT. Z .OR. X.LT.Y .AND. Y.LT.Z
```

Example 5: *Temperature Conversion: Convert temperatures from one unit into another using statement functions. Write a main program to test the functions based on a code. If the code is 1, convert from centigrade to Fahrenheit. If code is 2, convert from Fahrenheit to centigrade. Otherwise, print an error message.*

Solution:

```
      REAL FTEMP, CTEMP, TEMP, VALUE
      INTEGER CODE
C FUNCTION FTEMP CONVERTS FROM CENTIGRADE TO FAHRENHEIT
      FTEMP(TEMP) = TEMP * 9 / 5 + 32
C FUNCTION CTEMP CONVERTS FROM FAHRENHEIT TO CENTIGRADE
      CTEMP(TEMP) = (TEMP - 32) * 5 / 9
      READ*, CODE, VALUE
      IF (CODE .EQ. 1) THEN
          PRINT*, VALUE , ' C = ' , FTEMP(VALUE), ' F'
      ELSEIF (CODE .EQ. 2) THEN
          PRINT*, VALUE , ' F = ' , CTEMP(VALUE), ' C'
      ELSE
          PRINT*, 'INPUT ERROR'
      ENDIF
      END
```

The statement functions FTEMP and CTEMP convert the argument value to Fahrenheit and centigrade respectively. The statement functions are placed immediately after the declaration statements. The variables CODE and VALUE are read. Based on the value of CODE, the appropriate statement function is invoked and the converted value is printed.

4.5 Subroutine Subprograms

A function produces a single result. In many instances, we would like a subprogram to produce more than one result. Subroutines are designed to produce *zero*, *one* or *many* results. A *subroutine* consists of a subroutine header and a body.

Subroutines differ from functions in the following ways:

- A subroutine may return a single value, many values, or no value.
- To return results, the subroutine uses the argument list; thus, the subroutine argument list consists of *input* arguments and *output* arguments.
- Since the results are returned through arguments, a subroutine name is used for documentation purposes only and does not specify a value.
- The general form of the subroutine header is as follows:

```
      SUBROUTINE SNAME (a list of dummy arguments)
```

where

SNAME is the name of the subroutine; and

a list of dummy arguments is optional.

- A subroutine is called or invoked by an executable statement, the **CALL** statement. The general form of the statement is as follows:

```
      CALL SNAME (a list of actual arguments)
```

A subroutine is similar to a function in several ways. The subroutine actual and dummy arguments must match in type, number and order. At least one **RETURN** statement must be present to ensure transfer of control from a subroutine to the calling program.

Consider a program that consists of a subroutine and a main program. With each **CALL** statement in the main program, control is transferred to the subroutine. After the subroutine is executed, the **RETURN** statement ensures that control is transferred back to the calling program, to the statement immediately following the **CALL** statement.

4.5.1 *Examples on Subroutine Subprograms:*

Example 1: *Write a subroutine that exchanges the value of its two real arguments.*

Solution:

```
      SUBROUTINE EXCHNG(NUM1, NUM2)
      REAL NUM1, NUM2, TEMP
      TEMP = NUM1
      NUM1 = NUM2
      NUM2 = TEMP
      RETURN
      END
```

The subroutine EXCHNG can be invoked using the **CALL** statement. An example illustrating a call to the subroutine EXCHNG is given below:

Assume the variables X, Y are declared as real in the calling program and have the values 3.0 and 8.0 respectively. The **CALL** statement

CALL EXCHNG(X, Y)

after execution will exchange the value of X and Y. During the execution of the **CALL** statement, the value of actual argument X is passed to the dummy argument NUM1 and the value of actual argument Y is passed to the dummy argument NUM2. At this point, the execution control is transferred to the subroutine EXCHNG. The subroutine exchanges the values of variables NUM1 and NUM2. When the **RETURN** statement of the subroutine is executed, the control returns to the calling program and the new values of variables NUM1 and NUM2 are passed back to the actual arguments X and Y respectively. Therefore, the new value of variable X would be 8.0 and the value of variable Y would be 3.0.

Example 2: *Write a subroutine that takes three different integer arguments X, Y and Z and returns the maximum and the minimum.*

Solution:

```
      SUBROUTINE MINMAX(X, Y, Z, MAX, MIN)
      INTEGER X, Y, Z, MAX, MIN
      MIN = X
      MAX = X
      IF (Y .GT. MAX) MAX = Y
      IF (Y .LT. MIN) MIN = Y
      IF (Z .GT. MAX) MAX = Z
      IF (Z .LT. MIN) MIN = Z
      RETURN
      END
```

Examples illustrating calls to the subroutine MINMAX is given below:

Example 3: *Assume the variables A, B, C are declared as integer in the calling program and have the values 4, 6, 8 respectively. Also assume that MAX and MIN are integer variables. After the following **CALL** statement*

```
      CALL MINMAX(A, B, C, MAX, MIN)
```

is executed, the value of MAX will be 8 (the maximum of variables A, B, C) and the value of MIN will be 4 (the minimum of variables A, B, C). Note that the names of the actual arguments may be similar or different from the corresponding dummy arguments but the type must be the same.

Example 4: *If the following **CALL** statement*

```
      CALL MINMAX(C+4, -1, A+B, MAX, MIN)
```

*is executed, the value of MAX will be 12 and the value of MIN will be -1, since the first three actual arguments in the **CALL** statement are evaluated to 12, -1 and 10 respectively. Note here that the actual arguments can be expressions.*

Example 5: *Sum and Average: Write a subroutine to sum three integers and compute their average. The subroutine should return the sum and average of the three numbers. Write a main program to test the subroutine.*

Solution:

```
C     MAIN PROGRAM
      INTEGER X, Y, Z, TOTAL
      REAL AVERAG
      READ*, X, Y, Z
      CALL SUBSUM (X, Y, Z, TOTAL, AVERAG)
      PRINT*, 'TOTAL IS ', TOTAL
      PRINT*, 'AVERAGE IS ' , AVERAG
      END
C      SUBROUTINE SUBPROGRAM

      SUBROUTINE SUBSUM(A, B, C, TOTAL, AVG)
      INTEGER A, B, C, TOTAL
      REAL AVG
      TOTAL = A + B + C
      AVG = TOTAL / 3.0
      RETURN
      END
```

The subroutine SUBSUM has three dummy arguments A, B, C and returns two results, the value of the fourth argument TOTAL and the fifth argument AVERAG. The **CALL** statement in the main program invokes the subroutine.

Arguments X, Y, Z, TOTAL and AVERAG in the main program are the actual arguments. Note that, before the subroutine is called, arguments X, Y and Z have values and arguments TOTAL and AVERAG do not have a value. Arguments A, B, C, TOTAL and AVERAG in the subprogram are the dummy arguments. X, Y and Z are input arguments, TOTAL and AVERAG are output arguments.

The execution starts with the reading of variables X, Y and Z in the main program. The execution of the **CALL** statement transfers control to the subroutine SUBSUM. The value of the actual arguments X, Y and Z is passed to the dummy arguments A, B and C respectively. Since TOTAL and AVERAG in the main program are not initialized, no value is passed to the corresponding arguments in the subprogram. In the subroutine SUBSUM, execution begins with the first executable statement which computes the value of argument TOTAL. The next statement computes the average of the three arguments. The return statement returns control to the main program.

The values of arguments A, B, C, TOTAL and AVERAG in the subroutine are passed back to the arguments X, Y, Z, TOTAL and AVERAG in the main program respectively. The print statement in the main program prints the value of TOTAL and AVERAG, and the execution ends.

If the input to this program is

```
20, 60, 40
```

then the output is:

```
TOTAL IS 120
AVERAGE IS 40.0000000
```

Example 6: *Integer and Real Parts of a Number: The integer and decimal parts of a real number are to be separated. For example, if the number is 3.14, the integer part is 3 and the decimal part is 0.14. Write a subroutine SEPNUM to separate the real and integer parts.*

Solution:

```
C  SUBROUTINE SUBPROGRAM
      SUBROUTINE SEPNUM(NUMBER, IPART, RPART)
      REAL NUMBER, RPART
      INTEGER IPART
      IPART = INT(NUMBER)
      RPART = NUMBER - IPART
      RETURN
      END
C MAIN PROGRAM
      REAL NUMBER, PART2
      INTEGER PART1
      READ*, NUMBER
      CALL SEPNUM(NUMBER, PART1, PART2)
      PRINT*, ' INTEGER PART OF ', NUMBER, ' IS ', PART1
      PRINT*, ' DECIMAL PART OF ', NUMBER, ' IS ', PART2
      END
```

The subroutine has three dummy arguments: argument NUMBER represents the real number to be separated, argument IPART is the integer part of NUMBER and argument RPART represents the real part of the number.

If the input to this program is

```
57.231
```

then the output is:

```
INTEGER PART OF 57.2310000 IS 57
DECIMAL PART OF 57.2310000 IS 0.2310000
```

If the subroutine SEPNUM is invoked with the statement

```
 CALL SEPNUM(3.14, PART1, PART2)
```

then the value of PART1 is 3 and value of PART2 is 0.14.

4.6 Common Errors in Subprograms

There are several common errors that occur in the use of subprograms. We illustrate such errors through an example. The following program computes the new salary, given the current salary and the number of years of service. If the number of years is more than five, the salary is to be incremented by 8%, otherwise, the increment is 4%. The program uses a function INCSAL to compute the new salary. There are several errors in the program.

When the program is executed, the following error messages appear:

- *Error #1: INCSAL is an unreferenced symbol.* A function should return a single result stored in the function name. But in function INCSAL, the function name INCSAL is not assigned any value.
- *Error #2: Function INCSAL referenced as an integer but defined to be real.* The type of the function name in the main program is, by default, integer but its type in the function definition is real.

```
C   FUNCTION SUBPROGRAM
       REAL FUNCTION INCSAL(SALARY, YEARS)
       REAL SALARY, NSAL
       INTEGER YEARS
       IF (YEARS .GT. 5) THEN
          NSAL = SALARY * 8 / 100 + SALARY
       ELSE
          NSAL = SALARY * 4 / 100 + SALARY
       ENDIF
       END
C   MAIN PROGRAM
       REAL SALARY, YEARS
       READ*, SALARY, YEARS
       PRINT*, INCSAL(SALARY, YEARS)
       END
```

- *Error #3: Argument number 2 in call to INCSAL - real argument was passed but integer argument expected.* The type of argument number 2 in the calling statement does not match with its type in function subprogram. Mismatch of arguments is a common error in calls to both subroutines and functions.
- *Error #4: **RETURN** statement is missing.* The **RETURN** statement is missing in function INCSAL. This error may not be reported by many compilers.

4.7 Exercises

1. (a) Which of the following statement(s) is (are) FALSE?

   1. A function may contain more than one **RETURN** statement.
   2. A subroutine may return one value, many values, or no value.

3. A subroutine cannot call itself in FORTRAN.
4. The statement function is a non-executable statement.
5. A function may return more than one value.
6. A program may contain more than one subprogram.
7. A subroutine cannot call another subroutine.
8. The order and type of arguments in a subroutine call and the corresponding subroutine statement must be the same.
9. Use of subroutines increases the complexity of programming.
10. A function transfers results back to the calling program in the argument lists only.

2. What is printed by the following programs ?

```
1.     INTEGER A, B, X, Y, Z, F
       A = 2
       B = 3
       X = F(4, A)
       Y = B * 3
       Z = F(Y, X)
       PRINT*, X, Y, B, Z
       END
       INTEGER FUNCTION F(X,Y)
       INTEGER X, Y, Z
       Z = 2*Y
       F = X+Z
       RETURN
       END
```

```
2.     INTEGER OP
       REAL X, Y, CALC
       READ*, X, OP, Y
       PRINT*, CALC(X, OP, Y)
       READ*, X, OP, Y
       PRINT*, CALC(X, OP, Y)
       END
       REAL FUNCTION CALC(ARG1,OP,ARG2)
       INTEGER OP
       REAL ARG1, ARG2
       IF (OP .EQ. 1) THEN
           CALC = ARG1 + ARG2
       ELSEIF (OP .EQ. 2) THEN
           CALC = ARG1 - ARG2
       ELSE
           CALC = 0
       ENDIF
       RETURN
       END
```

Assume the input is

```
1.0,5,7.0
5.0,2,4.0
```

```
3.     LOGICAL  DIV
       INTEGER   N, J
       READ*,  N, J
       IF (DIV(N, J))THEN
           PRINT*,'YES'
       ELSE
           PRINT*,'NO'
       ENDIF
       END
       LOGICAL  FUNCTION  DIV(N, J)
       INTEGER  N, J
       DIV = N - N / J * J .EQ. 0
       RETURN
       END
```

Assume the input is

```
18  4
```

```
4.     INTEGER K , EVL
       K = 1
       PRINT*, EVL (K), K
       END
       INTEGER FUNCTION EVL (M)
       INTEGER M, K
       K = 2
       EVL = M * K
       RETURN
       END
```

```
5.     INTEGER A, B
       REAL FUN
       READ*, A, B
       A = FUN(A, B)
       B = FUN(B, A)
       PRINT*, FUN(A, B)
       END
       REAL FUNCTION FUN(X, Y)
       INTEGER X, Y
       FUN = X ** 2 + 2 * Y
       RETURN
       END
```

Assume the input is

```
1, 2
```

```
6.     INTEGER A, B, C, G
       G(A,B,C) = A * B-4 * C
       READ*, A, B, C
       PRINT*, G(A + B, B + C, C + A)
       END
```

Assume the input is

```
4 5 3
```

```
7.     LOGICAL F
       INTEGER X, Y, Z
       F(X, Y, Z) = X .GT. Y .AND. X .GT. Z
       READ*, X, Y, Z
       IF (F(X, Y, Z)) PRINT*, X
       IF (F(Y, X, Z)) PRINT*, Y
       IF (F(Z, X, Y)) PRINT*, Z
       END
```

Assume the input is

```
10 30 5
```

```
8.     INTEGER A,B,P,Q, G
       G(A,B) = A*A + B
       READ*, P, Q
       A = 1
       B = 2
       PRINT*, G(P,Q), G(Q,P), G(P+2, Q+2)*G(B,A)
       END
```

Assume the input is

```
 2  3
```

```
9.     LOGICAL FUNC
       INTEGER K, L
       FUNC(K,L) = K .GE. L
       READ*, K, L
       IF (FUNC(K,L)) THEN
           PRINT*, K
       ELSE
           PRINT*,L
       ENDIF
       END
```

Assume the input is

```
80  90
```

```
10.    INTEGER K, L
       K = -9
       L = 10
       PRINT*, MOD(ABS(K),L)
       END
```

```
11.    REAL A, B, DIST, X, Y
       DIST(X,Y) = SQRT(X ** 2 + Y ** 2)
       READ*, A, B
       PRINT*, DIST(A - 3.0, DIST(A, B) - 6.0)
       END
```

```
12.    INTEGER FUNCTION FUN(J, K, M)
       REAL SUM
       SUM = J + K + M
       FUN = SUM /3.0
       RETURN
       END
       INTEGER FUN,FUS, J, K
       FUS(J, K) = J * K / 2
       PRINT*, FUS(FUN(2, 3, 4), FUN(5, 6, 7))
       PRINT*, FUN(FUS(2, 3), FUS(4, 5), FUS(6, 7))
       END
```

Assume the input is

```
    6.0     8.0
```

```
13.    REAL F, G, A, B, X, Y
       F(A , B) = A + B
       G(X) = X ** 2
       READ*, Y
       PRINT*, G(Y), G(F(Y, Y + 2))
       END
```

Assume the input is

```
3.0
```

```
14.    LOGICAL  COMP
       REAL  X, Y, Z, A, B, C
       COMP(A, B, C) = A. GE. B .AND. A .GE. C
       READ*, X, Y, Z
       IF (COMP(X, Y, Z)) PRINT*, X
       IF (COMP(Y, X, Z)) PRINT*, Y
       IF (COMP(Z, X, Y)) PRINT*, Z
       END
```

Assume the input is

```
35.0   90.0   65.0
```

```
15.    INTEGER A,B,C
       A = 1
       B = 2
       C = 3
       PRINT*, A, B, C
       CALL CHANGE(A,B)
       PRINT*, A, B, C
       END
       SUBROUTINE CHANGE(A,B)
       INTEGER A,B,C
       C = B
       B = A + B
       A = C
       RETURN
       END
```

```
16.    INTEGER TOT
       REAL A, B
       A = 5.5
       B = 4.5
       CALL ADD(A,B,TOT)
       PRINT*, TOT
       END
       SUBROUTINE ADD(X,Y,SUM)
       INTEGER SUM
       REAL X, Y
       IF (X.LT.Y) THEN
       SUM = X + Y
       ELSE
       SUM = X - Y
       ENDIF
       RETURN
       END
```

```
17.    INTEGER JJ
       JJ = 1
       CALL TRY1(JJ,3)
       CALL TRY1(JJ,4)
       CALL TRY1(JJ,5)
       PRINT*, JJ
       END
       SUBROUTINE TRY1(X,Y)
       INTEGER X,Y,TRY2, N
       TRY2(N) = N-3
       X = TRY2(Y)+2*X
       RETURN
       END
```

```
18.    INTEGER X, Y, H
       H = 2
       CALL K(X,Y)
       PRINT*, H, Y, X
       END
       SUBROUTINE K(H,Y)
       INTEGER H,Y
       REAL X
       READ*, H, Y
       H = H / (Y+H)
       Y = H+3
       X = Y+2/3
       PRINT*, H, Y, X
       RETURN
       END
```

Assume the input is

```
   5 3 2
```

```
19.    REAL X,Y
       X = 3.0
       Y = 1.0
       CALL F(X,Y)
       PRINT*, X, Y
       END
       SUBROUTINE F(A,B)
       REAL A, B
       CALL G(B,A)
       B = A + B
       A = A - B
       RETURN
       END
       SUBROUTINE G(C,D)
       REAL C, D
       C = C + D
       D = C - D
       RETURN
       END
```

```
20.    INTEGER JJ
       JJ = 1
       CALL TEST1
       PRINT*, JJ
       END
       SUBROUTINE TEST1
       INTEGER JJ
       JJ = 2
       CALL TEST2
       RETURN
       END
       SUBROUTINE TEST2
       INTEGER JJ
       JJ = 3
       RETURN
       END
```

```
21.    REAL A, C
       A = 5
       CALL SUBPRO(A,C)
       PRINT*, A, C
       END
       SUBROUTINE SUBPRO(A,B)
       REAL A, B, C, X
       C(X) = X*2-2
       B = C(A)
       RETURN
       END
```

```
22.    SUBROUTINE CHANGE (W,X,Y,Z)
       INTEGER W,X,Y,Z
       W = X
       X = Y
       Y = Z
       Z = W
       RETURN
       END
       INTEGER A,B
       READ*, A, B
       CALL CHANGE(A * 2, B * 3, A, B)
       PRINT*, A * 2, B * 3
       END
```

Assume the input is

```
3    4
```

```
23.    INTEGER X, Y
       X = 3
       Y = X*3
       PRINT*, X, Y
       CALL CHANGE(X,Y)
       PRINT*, X, Y
       END
       SUBROUTINE CHANGE(X,Y)
       INTEGER X, Y
       X = X + 1
       Y = X - 1
       PRINT*, X, Y
       RETURN
       END
```

```
24.    LOGICAL FLAG
       REAL X, Y
       FLAG = .TRUE.
       READ*, X, Y
       CALL LOGIC (X, Y, FLAG)
       PRINT*, X, Y, FLAG
       END
       SUBROUTINE LOGIC (FLAG, X, Y)
       LOGICAL Y
       REAL X, Y
       IF (.NOT. Y) THEN
           FLAG = X**2+FLAG**2
           Y = .NOT. Y
       ELSE
           FLAG = (FLAG + X)
       ENDIF
       RETURN
       END
```

Assume the input is

```
4   5
```

```
25.    REAL A, B, C
       READ*, A,B
       CALL FIRST(A,B,C)
       PRINT*, A,B,C
       END
       SUBROUTINE FIRST (X,Y,Z)
       REAL X, Y, Z
       X = X + Y
       Y = Y - X
       CALL SECOND(X,Y,Z)
       RETURN
       END
       SUBROUTINE SECOND(N,M,L)
       REAL N,M,L
       L = THIRD(N,M)
       RETURN
       END
       REAL FUNCTION THIRD(J,K)
       REAL J,K
       THIRD = J - K
       RETURN
       END
```

Assume the input is

```
1   1
```

```
26.    INTEGER A, B
       LOGICAL FLAG
       READ*, A, B
       FLAG =  A .GT. B
       CALL SUB(A, B)
       PRINT*, A, B, FLAG
       END
       SUBROUTINE  SUB(A, B)
       INTEGER  A, B, T
       LOGICAL FLAG
       T = A
       A = B
       B = T
       FLAG = A .GT. B
       RETURN
       END
```

Assume the input is

```
6   3
```

```
27.    SUBROUTINE COMP (M , N)
       INTEGER M, N
       M = M + N
       N = M + N
       RETURN
       END
       INTEGER M, N
       READ*, M, N
       CALL COMP (M , N)
       PRINT*, M, N
       END
```

Assume the input is

```
1    2
```

```
28.   SUBROUTINE   MIDTERM (A, B)
      INTEGER  A, B, C
      IF (A . LT. B) THEN
          C = A
          A = B
          B = C
      ENDIF
      RETURN
      END
      INTEGER  A, B, C
      READ*, A, B, C
      PRINT*, A, B, C
      CALL  MIDTERM (B, A)
      PRINT*, A, B, C
      END
```

Assume the input is

```
17  23  31
```

```
29.   INTEGER B, C
      REAL A
      READ*, A,  C
      CALL BEST (A, REAL(C), B)
      PRINT*, A, B, C
      CALL BEST (A, B + 2.0 , C)
      PRINT*, A, B, C
      END
      SUBROUTINE BEST (ONE, TWO, THREE)
      REAL ONE, TWO
      INTEGER THREE
      THREE = ONE + TWO
      RETURN
      END
```

Assume the input is

```
9.5, 4
```

```
30.   REAL X, Y, A, B
      F(A, B) = A / B * 2
      CALL MYSUB(F(4.0, 1.0), X, Y)
      PRINT*, X, Y, F(X, X)
      END
      SUBROUTINE MYSUB (X, Y, Z)
      REAL X, Y, Z
      IF (X .LT. 0.0) THEN
          Z = X
      ELSEIF (X .EQ. 0.0) THEN
          Z = X + 2.0
      ELSE
          Z = X / 2.0
      ENDIF
      Y = Z * X
      RETURN
      END
```

```
31.   INTEGER NUM1, NUM2
      READ*, NUM1, NUM2
      CALL EXCHNG (NUM1, NUM2)
      PRINT*, NUM1, NUM2
      END
      SUBROUTINE EXCHNG (NUM1, NUM2)
      INTEGER NUM1, NUM2, TEMP
      LOGICAL COND
      IF (.NOT. COND(NUM1, NUM2)) THEN
          TEMP = NUM1
          NUM1 = NUM2
          NUM2 = TEMP
      ENDIF
      RETURN
      END
      LOGICAL FUNCTION COND(X, Y)
      INTEGER X, Y
      COND =X .GE. 0 .AND. Y .GT. X
      RETURN
      END
```

Assume the input is

```
3, -2
```

3. Which of the following functions may be used to find the maximum of two integer numbers K and M?

```
A.    INTEGER FUNCTION MAXA(K,M)
      INTEGER K, M
      MAXA = K
      IF (K.GT.M) MAXA = K
      RETURN
      END
```

```
B.    INTEGER FUNCTION MAXC(K,M)
      INTEGER K, M
      IF (M.GE.K) THEN
          MAXC = M
      ELSE
          MAXC = K
      ENDIF
      RETURN
      END
```

```
C.    INTEGER FUNCTION MAXB(K,M)
      INTEGER K, M
      MAXB = K
      IF (M.GT.K) MAXB = M
      RETURN
      END
```

4. Write a logical function subprogram FACTOR that takes two arguments and checks if the first argument is a factor of the second argument. Write a main program to test the function.
5. Write a function subprogram to reverse a three digit number. For example, if the number is 243, the function returns 342. Write a main program to test the function.
6. Write a function subprogram called AREA to compute the area of a circle. The argument to the function is the diameter of the circle. Write a main program to test the function.

7. Write a logical function subprogram that checks whether all its three arguments are non-zero. Write a main program to test the function.
8. Write the functions in problems 4, 5, 6, and 7 as statement functions.
9. Consider the following statement function IXX (J,K) = J-J/K*K. Which one of the following intrinsic (built-in) functions is the same as the function IXX ?

   i) MOD

   ii) MAX

   iii) MIN

   iv) SQRT
10. Rewrite the following function as a STATEMENT **FUNCTION**.

```
A.     REAL FUNCTION AREA(CIRCUM)
       REAL CIRCUM, RADIUS, PI
       PI = 3.14159
       RADIUS = CIRCUM/(2.0*PI)
       AREA = RADIUS **2*PI
       RETURN
       END
```

```
B.     REAL FUNCTION  X (A, B, C, D)
       Y = A ** 2 - B ** 2
       Z = C ** 3 + 1 / D ** 2
       X = Y / Z
       RETURN
       END
```

```
C.     REAL FUNCTION  AREA (R)
       AREA = 2 * 3.14 * R ** 2
       RETURN
       END
```

11. Write a function subprogram COST that computes the cost of postage according to the following: SR 0.50 for weight of less than an ounce, SR 0.10 for each additional ounce, plus a SR 50 extra charge if the customer wants fast delivery. The arguments to the function are the weight of the package and a logical variable FAST indicating fast delivery. Write a main program to test the function.
12. Write an function subprogram that takes the three sides of a triangle and returns the type of the triangle. For a right triangle, then the function returns an integer value 1; for an isosceles triangle, the value returned is 2; for an equilateral triangle, the function returns a value 3; otherwise, a value 0 is returned.
13. Which of the following functions return the maximum of the integers K, L and M?

```
I.     INTEGER FUNCTION F1(K,L,M)
       INTEGER K, L, M
       F = K
       IF (F .LT. L) F = L
       IF (F .LT. M) F = M
       F1 = F
       RETURN
       END
```

```
II.   INTEGER FUNCTION F2(K,L,M)
      INTEGER K, L, M
      IF (K .GE. L .AND. K .GE. M) THEN
          F2 = K
      ELSEIF (L .GE. M) THEN
          F2 = L
      ELSE
          F2 = M
      ENDIF
      RETURN
      END
```

```
III.  INTEGER FUNCTION F3(K,L,M)
      LOGICAL F4
      INTEGER K, L, M
      F4(K,L,M) = K .GE. L .AND. K .GE. M
      IF (F4(K,L,M)) F3 = K
      IF (F4(L,K,M)) F3 = L
      IF (F4(M,L,K)) F3 = M
      RETURN
      END
```

14. Given the following program which has some errors.

```
      INTEGER FUNCTION TEST (A, B)
      X = (A + B) ** 2
      Y = B * 2
      RETURN
      END
      REAL TEST
      PRINT*, TEST(1, 2, 3)
      END
```

Which of the following statements is correct?

I. Function name TEST is of type integer in function description but is a real in the calling program.

II. Function name TEST is not assigned a value in the function description.

III. Argument types do not match.

IV. The number of actual arguments is more than the number of dummy arguments.

15. Rewrite the following subroutine as a function subprogram.

```
      SUBROUTINE  DIVIDE (M, N, FACTOR)
      LOGICAL  FACTOR
      INTEGER M, N
      IF (N / M * M .EQ. N) THEN
          FACTOR = .TRUE.
      ELSE
          FACTOR = .FALSE.
      ENDIF
      RETURN
      END
```

16. Rewrite the following function subprogram as a subroutine. (Hint: The statement function is part of the function subprogram).

```
      REAL FUNCTION  SO (A, B, C)
      REAL A, B, C, FUN
      FUN (A, B, C) = A / B + C
      SO = FUN (A, B, C) / FUN (C, B, A)
      RETURN
      END
```

17. Write a subroutine that takes three arguments A, B, C and returns the arguments in increasing order. Write a main program to test the subroutine.
18. Write a subroutine that takes a numeric grade of a student and prints the letter grade based on the following policy:

numeric grade	letter grade
above 90	A
above 80	B
above 70	C
above 60	D
below 61	F

19. Write a subroutine that computes and returns the diameter, area, and the circumference of a circle given its radius.
20. Write the functions in problems 4, 5, 6, and 7 as subroutines.
21. Write a subroutine subprogram that takes the three sides of a triangle and prints one of the following types of the triangle: right triangle, isosceles triangle, or equilateral triangle.

4.8 Solutions to Exercises

Ans 1.

Statements 5, 7, 9 and 10 are FALSE.

Ans 2.

```
  8   9   3   25
0.0
    1.0
    NO
    2     1
53.0000000
44
30
7   11    21    5
90
9
5.0000000
```

9
11
9.0000000 64.0000000
90
1 2 3
2 3 3
1
12
0 3 3.0
2 3 0
-4.0 5.0
1
5.0 8.0
8 36
3 9
4 3
4 3
9.0 5.0 T
2.0 -1.0 3.0
3 6 T
3 5
17 23 31
17 23 31
9.5000000 13 4
9.5000000 13 24
32.0000000 4.0000000 2.0000000
-2 3

Ans 3.

b and c

Ans 4.

```
      LOGICAL FUNCTION FACTOR(AR1, AR2)
      INTEGER AR1, AR2
      IF (AR2 / AR1 * AR1 .EQ. AR2)THEN
          FACTOR = .TRUE.
      ELSE
          FACTOR = .FALSE.
      ENDIF
      RETURN
      END
C     MAIN  PROGRAM
      LOGICAL FACTOR
      INTEGER AR1, AR2
      READ*, AR1, AR2
      PRINT*, FACTOR(AR1, AR2)
      END
```

Ans 5.

```
      INTEGER N, REV
      READ*, N
      IF (N .GE. 100 .AND. N .LT. 1000)THEN
          PRINT*, REV (N)
      ELSE
          PRINT*, 'OUT OF RANGE'
      ENDIF
      END

      INTEGER FUNCTION REV(N)
      INTEGER N, K, J, M
      K = N / 100
      N = N - K * 100
      J = N / 10
      M = N - J * 100
      REV = M * 100 + J * 10 + K
      RETURN
      END
```

Ans 6.

```
      REAL FUNCTION AREA (D)
      REAL D, R
      R = D / 2
      AREA = R ** 2 * 3.14
      RETURN
      END
      REAL D
      READ*, D
      PRINT*, AREA(D)
      END
```

Ans 7.

```
      LOGICAL FUNCTION TEST(A, B, C)
      REAL A, B, C
      TEST = A .NE.0 .AND. B .NE. 0 .AND. C .NE. 0
      RETURN
      END
C     MAIN  PROGRAM
      LOGICAL TEST
      REAL A, B, C
      READ*, A, B, C
      IF (TEST(A, B, C)) THEN
          PRINT*, 'ALL NUMBERS ARE NON-ZERO'
      ELSE
          PRINT*, 'NOT ALL NUMBERS ARE NON-ZERO'
      ENDIF
      END
```

Ans 8.

```
      INTEGER AR1, AR2, REV
      LOGICAL FACTOR
      REAL AREA
      FACTOR(AR1, AR2) = AR2 / AR1 * AR1 .EQ.AR2
      REV(N) = (N - N / 10 * 10) * 100 +
     *(N - N / 100 * 100) / 10 * 10 + N / 100
      AREA (D) = (D / 2) ** 2 * 3.14
      TEST (A, B, C) = A.NE.0 .AND. B.NE.0 .AND. C.NE.0
```

Ans 9.

i

Ans 10.

```
A.    REAL AREA
      AREA(CIRCUM) = 3.14159 * (CIRCUM/(2.0 * 3.14159)) ** 2
```

```
B.    REAL X
      X(A, B, C, D) = (A ** 2 - B ** 2) / (C ** 3 + 1 / D ** 2)
```

```
C.    REAL AREA
      AREA(R) =  2 * 3.14 * R ** 2
```

Ans 11.

```
      REAL FUNCTION COST (WEIGHT, FAST)
      LOGICAL FAST
      IF (WEIGHT .LT. 1) THEN
          COST = 0.5
      ELSE
          COST = 0.5 + (WEIGHT - 1) * 0.10
      ENDIF
      IF (FAST) COST = COST + 50
      RETURN
      END
      LOGICAL FAST
      READ* , WEIGHT, FAST
      PRINT*, COST(WEIGHT, FAST)
      END
```

Ans 12.

```
      INTEGER FUNCTION TTYPE(A, B, C)
      REAL A, B, C
C   ASSUMING C IS THE LARGEST SIDE
      IF(SQRT(C) .EQ. SQRT(A + B)) THEN
         TTYPE = 1
      ELSEIF(A .EQ. B .AND. A .EQ. C) THEN
         TTYPE = 3
      ELSEIF(A .EQ. B .OR. B .EQ. C .OR. C .EQ. A) THEN
         TTYPE = 2
      ELSE
         TTYPE = 0
      ENDIF
      RETURN
      END
```

Ans 13.

I, II and III.

Ans 14.

I, II, III and IV.

Ans 15.

```
      LOGICAL FUNCTION FACTOR (M, N)
      INTEGER M, N
      IF (N / M * M .EQ. N) THEN
         FACTOR = .TRUE.
      ELSE
         FACTOR = .FALSE.
      ENDIF
      RETURN
      END
```

Ans 16.

```
      SUBROUTINE ANS(A,B,C,SO)
      REAL A, B, C, SO, FUN
      FUN (A, B, C) = A / B + C
      SO = FUN (A, B, C) / FUN (C, B, A)
      RETURN
      END
```

Ans 17.

```
      SUBROUTINE ORDER (A, B, C)
      INTEGER A, B, C, T
      IF (A .GT. B) THEN
          T = A
          A = B
          B = T
      ENDIF
      IF (A .GT. C) THEN
          T = A
          A = C
          C = T
      ENDIF
      IF (B .GT. C) THEN
          T = B
          B = C
          C = T
      ENDIF
      RETURN
      END
      INTEGER A, B, C
      READ* , A, B, C
      CALL ORDER (A, B, C)
      PRINT*, A, B, C
      END
```

Ans 18.

```
      SUBROUTINE LGRADE(MARK)
      REAL MARK
      IF(MARK .GE. 0  .AND. MARK .LE. 100) THEN
          IF(MARK .GT. 90) THEN
              PRINT*, 'A'
          ELSEIF(MARK .GT. 80) THEN
              PRINT*, 'B'
          ELSEIF(MARK .GT. 70) THEN
              PRINT*, 'C'
          ELSEIF(MARK .GT. 60) THEN
              PRINT*, 'D'
          ELSE
              PRINT*, 'F'
          ENDIF
      ELSE
          PRINT*, 'MARK OUT OF RANGE'
      ENDIF
      RETURN
      END
```

Ans 19.

```
      SUBROUTINE CIRCLE(R, D, A, C)
      REAL R, D, A, C
      D = R / 2
      A = 22.0 / 7.0 * R ** 2
      C = 2 *  22.0 / 7.0 * R
      RETURN
      END
```

Ans 20.

of problem 4

```
      SUBROUTINE   FACTOR (AR1, AR2, FLAG)
      INTEGER AR1, AR2
      LOGICAL FLAG
      FLAG =  AR2 / AR1 * AR1 .EQ. AR2
      RETURN
      END
```

of problem 5.

```
      SUBROUTINE   FIND (N, REV)
      INTEGER N, REV
      M = N / 100
      N = N - M * 100
      J = N / 10
      K = N - J * 10
      REV = K * 100 + J * 10 + M
      RETURN
      END
```

of problem 6.

```
      SUBROUTINE  CIRCLE(D, AREA)
      R = D / 2
      AREA = 22.0 / 7.0 * R ** 2
      RETURN
      END
```

of problem 7.

```
      SUBROUTINE   CHECK (A, B, C, TEST)
      LOGICAL TEST
      TEST = A .NE. 0 .AND. B .NE. 0 .AND. C .NE. 0
      RETURN
      END
```

Ans 21.

```
      SUBROUTINE   TTYPE (A, B, C)
      REAL A, B, C
C   ASSUMING C IS THE LARGEST SIDE
      IF(SQRT(C) .EQ. SQRT(A + B)) THEN
          PRINT* , 'RIGHT TRIANGLE'
      ELSEIF(A .EQ. B .AND. A .EQ. C) THEN
          PRINT* , 'EQUILATERAL TRIANGLE'
      ELSEIF(A.EQ.B .OR. B.EQ.C .OR. C.EQ.A)THEN
          PRINT* , 'ISOSCELES TRIANGLE'
      ELSE
          PRINT* , 'NONE OF THE OTHER TYPES'
      ENDIF
      RETURN
      END
```