Chapter 3

Principles of Programming Languages

1. If the `postfix` equivalent of the statement
`if c then x else y` is `cxy#`, then the `postfix` form `amn+mn-ab-#ba-#`
(a) has no syntactically valid prefix equivalent
(b) is equivalent to, `if a then m+n then if m-n else a-b else b-a`
(c) is equivalent to, `if a then if m+n then m-n then a-b else b-a`
(d) is equivalent to, `if a then m-n else if m+n then a-b else b-a`

2. A recursive function `f`, is defined as follows:

```
    f(n) = 2, if n = 0
         = m, if n = 1
         = 2xf(n-1) + 4xf(n-2), if n >= 2
```

If the value of `f(4)` is 88, then the value of m is
(a) `-1` (b) `0` (c) `2` (d) `1`

*3. Consider the FORTRAN statement - `DO5I = 1, 10`
To recognize `DO` as a keyword, the compiler (lexical analyzer) has to scan
(a) 5 characters beyond O (b) 3 characters beyond O
(c) no character beyond O (d) 8 characters beyond O

4. Use of recursion
(a) enhances logical clarity (b) makes debugging easier
(c) reduces execution time (d) reduces code size

***5.** A program has 100 instructions and another program (for the same problem) has 200 instructions. Which of the following comment logically follows?

(a) The execution time of the second program is more than that of the first.

(b) The execution time of the second program is same as that of the first.

(c) Compilation time of the second program, is more than that of the first.

(d) None of the above.

6. The conditional expansion facility of macro processors is provided to

(a) test a condition during the execution of the expanded program

(b) expand certain model statements depending upon the value of a condition during the execution of the expanded program

(c) implement recursion

(d) expand certain model statements depending upon the value of a condition during the execution of the macro expansion

***7.** Which of the following languages is case-sensitive (i.e., `IF` is not same as `if`)?

(a) FORTRAN (b) BASIC (c) C (d) None of the above

***8.** Val is a well known

(a) real-time language (b) object-oriented language

(c) command language (d) data-flow language

9. Consider the following pseudo-Pascal function

```
function fibo (n : integer) : integer;
begin
if(n = 0)         then fibo := 0
else if(n = 1)    then fibo := 1
else fibo := fibo(n-1)+fibo(n-2)
end
```

If `fibo(5)` is the function call, `fibo(1)` will be used

(a) 3 times (b) 4 times (c) 5 times (d) 6 times

***10.** An ordinary calculator treats all operators

(a) to be of equal precedence and associating to the right

(b) to be of equal precedence and associating to the left

(c) to be of unequal precedence and associating to the left

(d) in the usual mathematical sense

11. In which of the following parameter passing mechanisms, the actual argument has to be a variable?

(a) Pass by value (b) Pass by result

(c) Pass by value-result (d) Pass by reference

12. Which of the following class of statements usually produce no object code when compiled?

(a) Assignment (b) Declaration (c) Unreachable (d) Control

***13.** The principle that a function can always be replaced by its value (irrespective of the context) without changing the meaning is called

(a) referential transparency (b) orthogonality

(c) context-free (d) unbinding

14. Programming languages offer features to write functions to

(a) facilitate the implementation of top-down logic

(b) enhance logical clarity

(c) avoid programming across programs

(d) none of the above

15. The following pseudo-Pascal procedure

```
procedure palin;
var c   :char;
begin
     read(c);
     if NOT eoln then palin;
     write(c);
end
```

can be used to

(a) check if a given string is palindrome or not

(b) explain the concept of recursion

(c) reverse a given string

(d) delete a given line of text

***16.** In a certain language, the expression `5-3+2  x  4+1`, evaluates to `0`. Which of the following conclusions about the precedence and associativity of the operators +, −, * are correct?

(a) + has precedence over − and − has precedence over *

(b) All these have equal precedence and associate to the right

(c) All these have equal precedence and associate to the left

(d) + and − have equal precedence, which is over * and all associate to the left

17. The output of the following pseudo-Pascal program is

```
var a : integer;
procedure p;
 begin
  a := 2; write(a)
 end
begin
 a := 1; p; write(a)
end
```

(a) 2, 1 (b) 1, 2 (c) 2, 2 (d) 1, 1

18. Which of the following comparisons between static and dynamic type checking is incorrect?

(a) Dynamic type checking slows down execution.

(b) Dynamic type checking offers more flexibility to the programmers.

(c) Dynamic type checking is more reliable.

(d) Unlike static type checking, dynamic type checking is done during compilation.

19. The period of time between an allocation and its subsequent disposal is called

(a) scope (b) (dynamic) binding

(c) lifetime (d) longevity

20. Consider the following pseudo-Pascal program.

```
procedure  A;
  x,y : integer;
  procedure  B;
    x,z : real;
    statement 1
  end  B;
  procedure  C;
    i : integer;
    statement 2
  end  C;
end  A;
```

The variables accessible in statement 1 and statement 2 are

(a) `x` of `A`; `x,y` of `B`; `z` in `statement 1` and `x` of `B`; `y,i` in `statement 2`

(b) `x` of `B`; `y,z` in `statement 1` and `x` of `B`; `i,z` in `statement 2`

(c) `x` of `B`; `z,y` in `statement 1` and `x` of `A`; `i` and `y` in `statement 2`

(d) none of the above

21. Consider the following sequence of statements

```
Statement 1: A := B+C
Statement 2: D := A+C
Statement 3: E := A+B
Statement 4: G := D-E
Statement 5: H := E+A
Statement 6: I := H+G
```

Which of the statements can be executed in parallel?

(a) 2 and 4 (b) 4 and 5 (c) 5 and 6 (d) 4,5 and 6

22. If instructions are executed in parallel, whenever the required operands are available, then the execution time of the previous problem is logically same as that of sequentially executing

(a) 3 statements (b) 2 statements (c) 4 statements (d) 5 statements

23. Aliasing is a situation where

(a) two commands with different names share the same code

(b) a particular location is associated with more than one name
(c) different functions have the same name but require parameters of different types
(d) none of the above

24. Consider the following variant record declaration in pseudo-Pascal.

```
type abc = record
x : integer;
case y : integer of
 1 : (m : integer, n : real);
 2 : (e, f : integer);
end
```

Suppose a program uses an array of 'P' such records. Integer needs 2 bytes of storage and real r bytes. If the array occupies 480 bytes, the value of P will be

(a) 80 (b) 50 (c) 25 (d) 60

***25.** A recursive function f(x), is defined as follows:

```
if(x>100)
 return (x-10)
else return(f(f(x+11)))
```

For which of the following values of x, f(x) = 91?

(a) 100 (b) 91 (c) 1 (d) 101

***26.** English language uses full stop as a sentence

(a) separator (b) terminator
(c) separator and terminator (d) none of the above

***27.** In a hypothetical language, all operators have equal precedence and associate to the left. In this language, the expression 5 x 3 - 2 - 1 x 2 evaluates to

(a) 15 (b) 11 (c) 8 (d) 20

***28.** Overloading is

(a) functions having the same name but with different types of parameters
(b) a function used very frequently in a program
(c) an operator whose meaning is determined by the operand type
(d) all of the above

29. You are asked to use a computer to solve a problem given to you. How fast the computer solves your problem, depends on the

(a) algorithm used (b) language used for implementation
(c) programmer (d) computer

30. Which of the following is a dangling reference?

(a) Accessing a storage that is already disposed at the request of the user
(b) Accessing a storage that is already disposed at the request of the processor
(c) Accessing a variable that is declared but not initialized
(d) None of the above

31. Heap allocation is required for languages that
(a) support recursion
(b) support dynamic data structures
(c) use dynamic scope rules
(d) none of the above

32. Jensen's device makes explicit use of the property of
(a) value parameters
(b) reference parameters
(c) name parameters
(d) value-result parameters

33. For which of the following applications will you prefer a co-routine to a subroutine?
(a) Simulation of multi-processing (b) Complex searching process
(c) Handling inter-leaved lists (d) None of the above

34. Binding (of an identifier to a value) can occur while
(a) writing a program (b) compiling a program
(c) invoking a sub-program (d) executing a program

***35.** `COMMON` feature of FORTRAN is not found in most of the languages that followed it because
(a) it is difficult to implement
(b) memory is not of primary concern now-a-days
(c) virtual memory concept obviates it
(d) of its potential side-effects

***36.** Consider the following program fragment.

```
procedure exchange(A: integer, B: integer)
 temp : integer;
begin
 temp := A; A := B; B := temp
end;
begin
 M := 2; X[M] := 4;
 exchange(M, X[M]); write(M, x[2]);
end
```

If the parameters are passed by value, the output will be
(a) unpredictable (b) `2, 4` (c) `4, 2` (d) `2, 2`

***37.** `4, 2` will be the output of the previous question if the parameters are passed by
(a) reference (b) name (c) value (d) none of the above

***38.** If the parameters are passed by name, the output will be
(a) `2, 2` (b) `4, 4` (c) `2, 4` (d) `4, 2`

***39.** Choose the correct remarks that are based on the following pseudo-Pascal function.

```
function doit(x, y : integer) : integer;
begin
 if(x = 0) then doit := y
       else if (y = 0) then doit := x
               else doit := doit(x-1, y-1)
end
```

(a) It loops infinitely for some `x, y`.
(b) It doesn't work `if x` and `y` are both 0.
(c) Finds the greater of the two given non-negative integers.
(d) Finds the positive difference of two given non-negative integers.

***40.** Which of the following can be correctly identified to be Pascal tokens without look-ahead scanning?

(a) `:` (b) `:=` (c) `end` (d) `<`

***41.** Consider the following pseudo-Pascal program (assume (*starts a comment and*) ends a comment)

```
procedure doit(A, B, C) : integer;
begin
 B := B-2; C := A+C(*; write(C) *)
end;
var A, B : integer;
begin
  A := 10; B := 20; doit(A, A, A); write(A)
end
```

If this program prints `16`, then `A, B, C` should have been declared as

(a) all variable parameters (b) only `A` and `B` are variable parameters
(c) only `A` and `C` are variable parameters (d) only `B` and `C` are variable parameters

***42.** If the comment is removed in the previous problem, it will print `18, 8,` if the parameter declaration is

(a) only `A` and `B` are variable parameters (b) only `B` and `C` are variable parameters
(c) only `A` and `C` are variable parameters (d) all variable parameters

***43.** The vernacular language English can't be used as a Computer Programming language because

(a) it includes symbols that are not present in the keyboard
(b) it doesn't have a well-defined syntax
(c) it is ambiguous
(d) computers do not understand English

44. Choose the correct statements.

(a) In general, there is always an iterative equivalent of a recursive definition.
(b) Recursion and iteration are equally powerful.

(c) In iteration(unlike recursion), the body is carried out to completion each time, before the condition for termination is tested.

(d) Recursion is more powerful than iteration.

45. Binding cannot be done

(a) when separately compiled modules are being linked together

(b) during loading

(c) while writing a program

(d) none of the above

46. The target of an assignment statement should be

(a) 1-value (b) either 1-value or r-value

(c) r-value (d) none of the above

47. Which of the following problems are iterative, rather than recursive in nature?

(a) Simplex method for solving a linear programming problem.

(b) Newton-Raphson method for finding the roots of an equation.

(c) 8-Queen's problem.

(d) Depth first traversal of a given tree.

48. BNF is a meta-language for

(a) specifying the syntax of a language

(b) specifying a context free grammar

(c) describing how a program works

(d) shell programming

49. The basic difference between a procedural language and an applicative language is that the

(a) latter executes by evaluating expressions predominantly

(b) latter uses parameters, rather than assignment statements to communicate values

(c) former executes by evaluating expressions predominantly

(d) former uses parameters, rather than assignment statements to communicate values

***50.** The output of the following Pascal program is

```
program x;
var char, real : integer;
false : boolean;
begin
  char := 1; real := char; false := (char = real);
  if(false = true) then writeln('Don't Worry')
             else writeln('Be Happy')
end
```

(a) a compilation error message (b) `Don't Worry`

(c) a run time error message (d) `Be Happy`

Mr. Genius developed a language called Great, with the following instructions.

```
clr x - sets x to 0
inc x - increments x by 1
dec x - decrements x by 1
inv x - if x is non-zero, x will be set to 0.
        if x is zero, x will be set to 1.
```

The only control feature available is:

```
while x not 0
do
 statement list
end
```

The following 7 questions are based on this new language. Assume variables take the value 0 or any positive integer.

51. This language is as powerful as
(a) COBOL (b) LISP (c) C (d) C++

***52.** One of the four instructions is not needed. That is
(a) `clr x` (b) `inc x` (c) `dec x` (d) `inv x`

53. The program

```
clr C; clr B;
while (A not 0)
do
   inc C; dec A;
end;
while(C not 0)
do
   inc A; inc B; dec C;
end
```

(a) transfers the contents of A to B (b) copies the contents of A to B
(c) copies the contents of A to B and C (d) transfers the contents of A to C

54. The program

```
clr C;
while A not 0
   do
      clr x;
      while (B not 0)
      do
         inc C; inc x; inc B;
      end;
      while (x not 0)
         do
            inc B; dec x;
         end;
      dec A;
end
```

(a) computes A+B (b) computes A−B (c) computes A × B (d) computes A × A

55. If we add one more instruction `cpy x, y` – which copies the contents of x to y, then the following program,

```
clr C; cpy A, x;
while(x not 0)
do
   cpy x, y;
   while(y not 0)
   do
       inc C; dec y;
   end;
   dec x;
end
```

(a) finds 1+2+3+...+A
(b) computes A+A
(c) computes A*A
(d) computes A+x

56. The following program

```
cpy A, x; clr C;
while(x not 0)
do
    inv C; dec x;
end
```

(a) assigns 0 to C
(b) assigns 1 to C
(c) assigns 0 to C, if A is even, else assigns 1 to C
(d) assigns 0 to C, if A is odd, else assigns 1 to C

57. The following program

```
cpy A, x;
while(x not 0)
do
    inv M; clr x;
end;
cpy A, x; inv x;
while(x not 0)
do
    inc M; clr x;
end
```

(a) computes M+x
(b) does not change the value of M
(c) does the same thing as – `if A not 0 then inv M else inc M`
(d) does the same thing as – `if A not 0 then inc M else inv M`

58. In which of the following cases, is it possible to obtain different results for call-by-reference and call-by-name parameter passing?

(a) Passing an expression as a parameter (b) Passing an array as a parameter
(c) Passing a pointer as a parameter (d) Passing an array element as a parameter

The next two questions are based on the following program segment in pseudo-Pascal.

```
var x, y : integer;
procedure A(var z : integer);
var x : integer;
begin
 x := 1; B; z := x;
end;
procedure B;
begin
 x := x-1;
end;
begin
 x := 5; A(y); write(y);.
end
```

59. If the language uses static scope rules, the output will be

(a) 0 (b) 3 (c) 4 (d) 5

60. If the language uses dynamic scope rules, the output will be

(a) 0 (b) 3 (c) 4 (d) 5

61. What will the following function compute?

```
function what(x, n : integer) : integer;
var
  value : integer;
begin
  value := 1;
  if(n > 0) then
  begin
      if(n mod 2 = 1) then
      value := value*x;
      value := value*what(x*x, n div 2);
  end;
  what := value;
end;
```

(a) x+n (b) x*n (c) x*x (d) x^n

62. An array A consists of *n* integers in locations `A[0], A[1], ... A[n-1]`. It is required to shift the elements of the array cyclically to the left by k places, where `1 ≤ k ≤ n - 1`. An incomplete algorithm for doing this in linear time, without using another array is given below. Complete the algorithm by filling in the blanks. Assume all variables are suitably declared.

```
min := n; i := 0;
while (________) do
begin
 temp := A[i]; j := i;
 while (_______) do
 begin
 A[j] := _______; j := (j+k) mod n;
 if(j < min) then
     min := j;
end;
A[(n+i-k) mod n] := ________;
i := ________;
end;
```

(a) `i > min; j != (n+i) mod n; A[j+k]; temp; i+1;`
(b) `i < min; j != (n+i) mod n; A[j+k]; temp; i+1;`
(c) `i > min; j != (n+i+k) mod n; A[j+k]; temp; i+1;`
(d) `i < min; j != (n+i-k) mod n; a[(j+k) mod n]; temp; i+1;`

The next two questions are based on the following program.

```
program main;
var r: integer;
procedure two;
begin write(r)  end;
procedure one;
var r: integer;
begin r := 5; two; end;
begin
  r := 2; two; one; two;
end
```

63. If static scoping is used by all variables, the output will be
(a) 222 (b) 255 (c) 252 (d) 555

64. If dynamic scoping is used by all the variables, the output will be
(a) 222 (b) 255 (c) 252 (d) 555

65. Consider the recursive function

```
function fib(n : integer) : integer;
begin
 if(n = 0) or (n = 1) then fib := 1
 else fib := fib(n-1)+fib(n-2)
end;
```

The function is run on a computer with a stack of size 64 bytes. If only the `return` address and parameter are passed to the stack and they need two bytes each, estimate the maximum value of `n` for which the stack does not overflow.

(a) 4 (b) 6 (c) 10 (d) 9

66. FORTRAN does not permit recursion because

(a) it uses static allocation for storing variables
(b) it uses dynamic allocation for storing variables
(c) stacks are not available in all machines
(d) it is not possible to implement recursion on all machines

67. A data driven machine is one that executes an instruction if the needed data is available. The physical ordering of the code listing does not dictate the course of execution. Consider the following pseudo-code.

(A) Multiply E by 0.5 to get F
(B) Add A and B to get E
(C) Add B with 0.5 to get D
(D) Add E and F to get G
(E) Add A with 10.5 to get C

Assume A, B, C are already assigned values and the desired output is G.
Which of the following sequence of execution is valid?

(a) B, C, E, A, D (b) C, B, E, A, D
(c) A, B, C, D, E (d) E, D, C, B, A

***68.** In the previous question, in how many different ways can the 5 instructions be sequenced?

(a) 10 (b) 8 (c) 6 (d) 12

***69.** In a demand-driven machine, an instruction is not executed until its output is needed. For the previous question, in what order will the instructions be sequenced?

(a) D, B, A, C, E (b) A, B, C, D, E
(c) E, D, C, B, A (d) None of the above

70. Choose the correct statements.

(a) Step-wise refinement uses top-down methodology
(b) Step-wise refinement uses bottom-up methodology
(c) Use of library routines facilitate bottom-up methodology
(d) None of the above

71. The following is an incomplete pseudo-Pascal function to convert a given decimal integer (in the range -8 to 7) into a binary integer in 2's complement form. Determine the expressions that complete the program.

```
function TWOCOMP(N : integer) : integer;
var
  REM, EXPO, BINARY : integer;
begin
 if(n ≥ - 8)  and  (N ≤ 7)   then
begin
 if N < 0 then
    N := ...;
    BINARY := 0; EXPO := 1;
    while N <> 0 do
          begin
          REM := N mod 2;
          BINARY := BINARY + ... * EXPO;
          EXPO := EXPO*10;
          N := ...;
          end;
     TWOCOMP := BINARY
  end
 end;
```

(a) N+1; REM; N div 2 (b) N+16; REM; N div 2
(c) N+1; REM; N mod 2 (d) N+16; REM; N mod 2

***72.** In the following pseudo-Pascal program segment, the value x, after the execution of the program is

```
X := -10;
y := 20;
if X>Y
then
    if X<0
    then X := abs(X)
    else X := 2*x;
```

(a) 20 (b) 30 (c) –10 (d) –20

73. Consider the following macro definition.

```
macro  Add x, y
       Load y
       Mul x
       Store y
end macro
```

x and y are

(a) variables (b) identifiers (c) actual parameters (d) formal parameters

***74.** Which of the following strings can definitely be said to be tokens without looking at the next input character while compiling a Pascal program.

I. `begin` II. `program` III. `<>`

(a) I only (b) II only (c) III only (d) All of the above

75. Assume X and Y are non-zero positive integers. The following pseudo-Pascal program

```
while X <> Y do
 if X > Y then X := X-Y
          else Y := Y-X:
write(X);
```

(a) computes the LCM of two numbers

(b) divides the larger number by the smaller number

(c) computes the GCD of two numbers

(d) finds the smaller of two numbers

76. The value of X printed by the following pseudo-Pascal program is

```
program COMPUTE(input, output);
var
 X : integer;
procedure FIND(X : real);
begin
 X := sqrt(X);
end;
begin
 X := 2; FIND(X); writeln(X)
end.
```

(a) 2 (b) $\sqrt{2}$ (c) Run-time error (d) none of the above

***77.** A language with string manipulation facilities uses the following operations.

`head(s)` - returns the first character of string `s`.

`tail(s)` - returns all but the first character of string `s`.

`concat(s1, s2)` concatenates strings `s1` and `s2`

The output of `concat(head(s), head (tail(tail(s))))`, where `s` is `acbc`, is

(a) `ab` (b) `ba` (c) `ac` (d) `aa`

***78.** Which of the following statements are true?

I. As the number of entries in the hash table increases, the number of collisions increases.

II. Recursive programs are efficient.

III. The worst case complexity of Quick sort is $O(n^2)$.

IV. Binary search using a linear linked list is efficient.

(a) I and II (b) II and III (c) I and IV (d) I and III

***79.** Consider the following high level program segment.

```
var
 A, B, W, X, Y  : unsigned byte;
 Z: unsigned integer;
begin
 X  := A+B;
 Y  := abs(A-B);
 W  := A-B;
 Z  := A*B;
end
```

Assuming `integer` occupies 2 bytes and the initial values of `A` and `B` are `5CH` and `92H` respectively, the final values of `W, X, Y` and `Z` will be

(a) `CAH, EEH, 36H, 3478H` (b) `AH, EEH, 36H, 3478H`
(c) `AH, EBH, 36H, 3478H` (d) `CAH, EBH, 36H, 3478H`

***80.** Consider the following pseudo-Pascal function, where `A` and `B` are non-zero positive integers. What is the value of `GET(3,2)`?

```
function GET(A, B : integer)  : integer;
begin
  if B = 0
  then
    GET := 1
  else if A < B
        then
          GET := 0
        else
          GET := GET(A-1,B) + GET(A-1,B-1)
end;
```

(a) 1 (b) 2 (c) 3 (d) 7

***81.** A variant record in Pascal is defined as

```
type varirec = record
                    number: integer;
                    case (var1, var2) of
                      var1 : (x, y : integer);
                      var2 : (P, q : real)
                    end
               end
```

Suppose an array of 100 such records was declared on a machine, which uses 4 bytes for an integer and 8 bytes for a real. How much space would the compiler have to reserve for the array?

(a) 2800 (b) 2400 (c) 2000 (d) 1200

Let the symbol D stand for a variable that is defined or refined. Let the symbol K stand for a variable that is killed. Let the symbol U stand for a variable that is used. The next four questions are based on the above notations.

82. Consider the assignment statement

```
var1 = var1 + var2;
```

The sequence of notations that correctly reflects the usage of the variable `var1` is

(a) KD (b) D (c) UD (d) UKD

83. Which of the following sequences (for a particular variable) is harmless but suspicious?

(a) DU (b) KD (c) UU (d) DD

84. Which of the following sequences (for a particular variable) are probably bugs?

(a) KK (b) UK (c) KU (d) DK

85. In addition to the above notations, let us use the notation –X, where X is one of D, K or U, to mean that nothing of interest (concerning the variable under consideration) happened to the variable. Which of the following situations are probably anomalous?

(a) –U (b) –D (c) –K (d) None of the above

86. Vernacular languages (like English) and programming languages have a lot of similarities. In a broad sense, the nouns and verbs are comparable to

(a) operators and identifiers respectively (b) operands and identifiers respectively
(c) operands and operators respectively (d) operators and functions respectively

87. Consider the following C program.

```
#include "stdio.h"
main( )
{
 enum boolean (true, false);
 enum boolean a, b, c;
 a = b = true;
 c = (a==b);
 if (c == a)
     printf("TRUE");
 else
     printf("FALSE");
}
```

The output of the above program will be

(a) TRUE (b) FALSE (c) computer dependent (d) unpredictable

Answers

1. b	2. d	3. a	4. a, d	5. d
6. d	7. c	8. d	9. c	10. a
11. b, c, d	12. d	13. a	14. a, b, c	15. b, c

16. a, d	17. c	18. a, b	19. c	20. c
21. a, b	22. c	23. a, b	24. d	25. a, b, c, d
26. b	27. a	28. a, c	29. a, b, c, d	30. a, b, c
31. b	32. c	33. a, b, c	34. a, b, c, d	35. b, c, d
36. b	37. a	38. b	39. a, d	40. b
41. a	42. a	43. c	44. a, b, c	45. d
46. a	47. a, b	48. a, b	49. a, b	50. b
51. a, b, c, d	52. a	53. b	54. c	55. a
56. c	57. c	58. a	59. c	60. a
61. d	62. d	63. a	64. a	65. b
66. a	67. a, b	68. d	69. d	70. a, c
71. b	72. c	73. d	74. c	75. c
76. a	77. a	78. d	79. a	80. c
81. c	82. d	83. d	84. a, c, d	85. a, c
86. c	87. b			

Explanations

3. Had the statement been `DO5I = 1.10,` `DO` would not have been a keyword, but a prefix of the token `DO5I`. So, the compiler has to scan till `','` to make sure `DO` is used as a keyword.

5. There is no relationship between code size and execution time. For example, use of recursion, generally results in compact code, but execution time will be more.

7. As an implication of this, `int if;` is an illegal C declaration, but `int IF;` is legal.

8. Data-flow languages, uses the availability of information rather than the logical or physical ordering of instructions in a program to decide whether an instruction is to be executed. So, the tenth instruction may get executed before the sixth if the information needed for the execution of the tenth instruction is available before the information needed for the sixth is available. This way, data-flow languages exploit the inherent parallelism in a particular program.

10. APL also evaluates an arithmetic expression, the same way as a calculator.

13. Orthogonality is the principle that each component of a language should be independent of the other components. Context-free in a broad sense means replacement of one pattern by another, irrespective of the context.

16. In option (a), the expression is equivalent to
`(5 - (3 + 2) × (4 + 1))`, which evaluates to `0`.
In option (b), it is `((((5 - 3) + 2) × 4) + 1)`, which yields `17`.
In option (c), it is `(5 - (3 + (2 × (4 + 1))))`, which evaluates to `8`.
In option (d), it is `((5 - (3 + 2)) × (4 + 1)`, which evaluates to `0`.

25. In fact `f(x)` will be `91` for any `x` less than 101. Let us use the short form $f^3(x)$, to denote `f(f(f(x)))`. What will be the value of $f^n(91)$?
$f^n(91) = f^{n+1}(102) = f^n(92) = f^{n+1}(103)$ etc.,
At some stage, it will be $f^n(100) = f^{n+1}(111) = f^n(101) = f^{n-1}(91)$. We started with $f^n(91)$ and reduced it to $f^{n-1}(91)$. This n–l can be reduced n–2, etc., Ultimately one gets $f^1(91)$, which can be proved to be `91`, using the same logic.
What is left to be proved is, if you start with any number less than 101, you should be getting $f^n(91)$, for some *n*. This is obvious because we keep on adding 11, till the argument falls in the range 101 to 111. Afterwards, a 111 becomes 91 in two steps - 101 becomes 91 in one step. For any other number in this range, we subtract 10 and in the next step, add 11. The net effect is adding 1 to the number. Progressing this way, one gets 111, which will be reduced to 91 in two steps.

26. There are two views regarding the use of semi-colon in the programming languages. Some languages (like PL/I) use it as a terminator while some languages (like Pascal) use it as a separator. In English it is used as a terminator, for the simple reason—if it is a separator, the last sentence should not end with a full stop.

27. In this language, the given expression is equivalent to `(5 × (3 - (2 - (1 × 2))))`, which evaluates to `15`.

28. `+` is an operator that can act on operands of different types. The actual addition process depends on the type. So, `+` is said to be overloaded. Similarly, a function like `write` can take arguments of different types. So, it is also overloaded.

35. FORTRAN was developed at a time when large memories were expensive. So, to make optimal use of available memory, the idea of many variables sharing the same space was supported, in spite of the inherent dangers. This is the reason why FORTRAN doesn't support the space consuming recursion technique.

36. If the parameters are passed by value, the function will be manipulating a local copy of the argument value. Any change will be local to the function and hence will not be reflected in the calling environment.

37. In call by reference, the address of the actual arguments will be passed to the function. Any change done inside the function will be reflected outside the function also.

38. In this case, the following statements will be executed by the function. `temp: = M; M := X[M]; X[M] := temp;` So, what is evaluated is `temp := 2; M := X[2]; X[M] := temp;` i.e., `M` will be assigned `4`, after which `X[M]`, i.e., `X[4]` will be assigned `2`. `X[2]` remains unaltered. So, `4, 4` will be printed.

39. If `x` and `y` are negative integers, this loops infinitely resulting in abnormal termination because of stack overflow. If `0,0` is the input `0` will be returned.

40. `end` may be prefix of an identifier.
`<`, if followed by `=`, can't be treated as a single entity.
`:`, if followed by `=`, can't be treated as a single entity.

41. If A, B, C are all declared to be variable parameters, the call `doit(A, A, A)` executes the statements - `A:=A-2; A:=A+A;` i.e., it evaluates `A:=10-2; A:=8+8;` Because of the variable parameter declaration, these changes will be global. Hence `16` will be printed.

42. Refer Qn. 41. If `A` and `B` are declared as variable parameters, what is evaluated is `A:=10-2; A:=8+10;` This `18` will have no impact outside as it is assigned to `C`, a local variable. So, it prints `18` and `8`.

43. A word may have more than one meaning. We use our common sense to decide in what sense it is used in the current context which a computer can't.

50. In Pascal, pre-defined meaning of the standard identifiers can be changed by the user. Here the standard identifiers `false, char, real` are used as user-defined variables.

52. `clr x` can be simulated as :

```
while x not 0
do
   dec x
end.
Another way is :
inc x
inv x
```

68. Instructions `B, C, and E` can be executed in any order. Instruction `A` can be executed after executing instruction `B`. So, 12 possible ways.

69. Instruction `C` and instruction `E` will never get executed as their output is not needed.

72. Since the condition `X > Y` fails, execution takes the `else` route, which does nothing. So the value of `X` will remain unchanged.

74. `beginabc` is a single token representing an identifier. Just after scanning the symbols `b, e, g, i, n,` — it should not be incorrectly grouped as the keyword `begin`. For similar reasons, program cannot be grouped as a keyword, without scanning the delimiter. But, `<>` can be grouped as a Boolean operator, does not matter what symbol is following the `>`.

77. `tail(s)` returns `cbc`. `tail(tail(s))` returns `bc`. So the given operation reduces to `concat(head(acbc),head(bc))`, which is `concat(a,b)` - which is the string `ab`.

78. `Recursive` program takes more time than its equivalent non-recursive version and so is not efficient. This is because of the function call overhead.

In binary search since every time, the current list is probed at the middle, random access is preferred. Since linked list does not support random access, binary search implemented this way is inefficient.

79. The values of the variables `W, X, Y` and `Z` after the execution of the program will be `CAH, EEH, 36H,` and `3478H`.

80.
```
GET (3,2) = GET(2,2) + GET(2,1)
          = GET(1,2) + GET(1,1) + GET(1,1) + GET(1,0)
          = 0 + GET(1,1) + GET(1,1) + 1
          = 2 × (GET(0,1) + GET(0,0)) + 1
          = 2 + 1 = 3.
```

81. `number` field is compulsory. This needs 4 bytes. `var1` needs 4 + 4 = 8 bytes of memory. `var2` needs 8 + 8 = 16 bytes of memory. So the compiler will allocate max(8, 16) = 16 bytes. So, totally 4 + 16 = 20 bytes. To store 100 such records, one needs 2000 bytes of memory.