UNIT–4

PROGRAMMING

UNIT–4

PROGRAMMING

4.1 C–FUNDAMENTALS

LEVEL–1

Q.1 By default any real number in 'C' is treated as
(a) a float
(b) a long double
(c) depends upon the memory model that you are using
(d) a double

Q.2 Trace the output:

```
main ()
{
print ("%c",abcdefgh[4]);
}
```

(a) abcdefgh
(b) Error
(c) e
(d) d

Q.3 The declarations

```
typedef float ht[100];
ht men, women;
```

defines _____
(a) men and women as floating point variables
(b) ht, men and women as floating point variables
(c) men and women as 100 element floating point array
(d) none of these

Q.4 If y is of integer type, then the expressions
3 * (y – 8) / 9 and (y – 8) / 9 * 3
(a) may or may not yield the same value
(b) must yield the same value
(c) must yield different values
(d) none of the above

Q.5 The difference between rand( ) and srand( ) is _____
(a) Type of rand( ) function is int and srand( ) is void
(b) There is no difference between rand( ) and srand( ) both do same function
(c) The srand( ) function is used to initialize the random number generator whereas rand( ) function returns a random positive integer.
(d) both (a) and (c)

Q.6 To scan a and b given below which scan f( ) statement would you use?

```
float a;
    double b;
```

(a) scanf("%f%lf", &a, &b);
(b) scanf("%f%Lf", &a, &b);
(c) scanf("%Lf%Lf", &a, &b);
(d) scanf("%f%f", &a, &b);

Q.7 Maximum combined length of the command line arguments including the spaces between adjacent arguments is

(a) It may vary from one operating system to another

(b) 67 characters

(c) 256 characters

(d) 128 characters

Q.8 The following program fragment

```
int k = – 7 ;
printf("%d", 0 < !k};
```

(a) prints an unpredictable value

(b) is illegal

(c) prints a non-zero value

(d) prints 0

Q.9 Which of the following are not keywords in C?

(a) printf

(b) main

(c) IF

(d) none of the above.

Q.10 If variable can take any integral values from 0 to n, where n is a contant integer, then the variable can be represented as a bits field whose width is the integral parts of–

(a) $\log_2(n)+1$

(b) $\log_2(n+1)+2$

(c) $\log_2(n-1)+1$

(d) None of these

Q.11 The following lines, if included in a program, will cause one of the followings errors. Indicate the correct one.

```
{
    double c;
    scanf("%c",c);
```

(a) Runtime error

(b) Compilation error

(c) Typedef error

(d) No error

Q.12 What will be the output of the following program?

```
#include <stdio.h>
int main()
{
    char * s1;
    char far * s2;
    char huge * s3;
    printf ("%d%d\n", sizeof(s1), sizeof(s2), sizeof(s3));
    return 0;
}
```

(a) 442 in TC/C++ compiler

(b) error in code

(c) error in VC++ or gcc compiler

(d) None of the above

Q.13 Which of the following is the correct output for the program given below?

```
#include <stdio.h>
void fun (int);
int main ()
{
int a ;
a = 3;
fun (a);
printf (:\n");
return 0;
}
void fun (int n)
{
    if (n > 0)
    {
        fun (–n);
        printf (:%d", n);
        fun (–n);
    }
}
```

(a) 0 2 1 0

(b) 1 1 2 0

(c) 0 1 0 2

(d) 0 1 2 0

Q.14 What do you mean by a translation unit?

(a) a set of source files seen by the compiler and translated as a unit.

(b) a set of linkers for compiler and translated as a unit.

(c) is a database that stores so called "segments".

(d) Is a language interface provider.

Q.15 Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main()
{
    printf("%d%d\n", 32<<1, 32<<0);
    printf("%d%d\n", 32<<-1, 32<<-0);
    printf("%d%d\n", 32>>1, 32>>0);
    printf("%d%d\n", 32>>-1, 32>>-0);
    return 0;
}
```

(a) garbage values

(b) 64 32
0 32
16 32
0 32

(c) all zeros

(d) 8 0
0 0
32 0
0 16

Q.16 C is a

(a) high level language

(b) low level language

(c) high level language with some low level features.

(d) low level language with some high level features.

Q.17
```
#include <stdio.h>
int main ()
{
    auto int i = 100;
    printf ("i = %d\n", i);
    {
        int i = 1;
        printf ("i = %d\n", i);
        {
            i += 1;
            printf ("i = %d\n", i);
        }
        printf ("i = %d\n", i);
    }
    printf ("i = %d\n", i);
    return 0;
}
```

(a) i = 100
i = 1
i = 2
i = 2
i = 2

(b) i = 100
i = 100
i = 101
i = 101
i = 101

(c) i = 100
i = 1
i = 2
i = 2
i = 100

(d) i = 100
i = 100
i = 101
i = 101
i = 100

Q.18 Even if a particular implementation doesn't limit the number of characters in an identifier, it is advisable to be concise because
(a) chances of typographic errors are less
(b) it may be processed by assembler, loaders, etc. which may have their own rules that may contradict the language rules
(c) by being concise, one can be mnemonic
(d) None of the above

Q.19 Coercion
(a) takes place across an assignment operator.
(b) takes place if an operator has operands of different data types.
(c) means casting
(d) none of the above

Q.20 Which of the following comments are true?
(a) C provides non input-output features.
(b) C provides no file access features.
(c) C borrowed most of its ideas from BCPL.
(d) C provides no features to manipulate composite objects.

Q.21 Which of the following comments about wide characters is/are true?
(a) It is the binary representation of a character in the extended binary set.
(b) It is of integer type wchart t.
(c) End of file is represented by WEOF.
(d) None of the above.

Q.22 int i = 5;
is a statement in a C program. Which of the following are true?
(a) during execution, value of i may change but not its address.
(b) during execution both the address and value may change
(c) repeated execution may result in different addresses for i
(d) i may not have an associated address

Q.23 C preprocessor
(a) takes care of conditional compilation
(b) takes care of macros
(c) takes care of include files
(d) acts before compilation

Q.24 The statement printf("%d", 10?0?5:11:12);
prints
(a) 10
(b) 0
(c) 12
(d) 11

Q.25 What are the files which are automatically opened when a C file is executed?
(a) Stdin
(b) Stdout
(c) Stderr
(d) All of the above

Q.26 The fields in a structure of a C program are by default
(a) private
(b) public
(c) protected
(d) none of the above

LEVEL–2

Q.27
```
main ( )
{       int a =6, b =10, x;
x=a & b;
printf("%d", x);
}
```
find the value of x.
(a) 2
(b) 10
(c) 5
(d) 7

Q.28 What will be the values of a, b and c after execution?
```
int a, b, c;
b = 10;
c = 15;
a = ++b + c++;
```
(a) a = 27, b = 11, c = 16
(b) a = 26, b = 11, c = 16
(c) a = 25, b = 10, c = 15
(d) a = 27, b = 10, c = 15

Q.29 Consider a following declaration

```
main ( )
    {
    int i, n;
    float p;
        scanf("%d%d%f",&i, &n, &p);
        print("%f", p*pow((1+i), n));
    }
```

select the correct statement

(a) The above program is also use for $(1 + i^1 + i^2 + i^3 + \ldots + i^n)^n$

(b) The above program is use for finding $(a + (a + 1)^1 + (a + 2)^2 + \ldots\ldots)$

(c) The above program is use for finding compund interest.

(d) The above program is use for finding the fibbonacci series with n numbers and n power.

Q.30 Consider a following declaration:

```
int i = 8, j = 5,
float x = 0.005, y = -0.01;
char c = 'c', d = 'd';
f = 2 * ((i/5) + (4 * (j – 3))%(i + j – 2));
```

Then, the data type of f is _____ and value is _____

(a) float, 18

(b) int, 13

(c) int, 18

(d) float, 13

Q.31 What is the output of the following 'C' program?

```
main ( )
{
    int i = 2;
    printf("%d%d", ++i, ++i);
    }
```

(a) 4 4

(b) 3 4

(c) 4 3

(d) Output may vary from compiler to compiler

Q.32 In the following 'C' code in which order the functions would be called?

```
a = (f1 (23, 14) * f2 (12/4)) + f3 ( );
```

(a) None of these

(b) f3, f2, f1

(c) f1, f2, f3

(d) The order may vary from compiler to compiler

Q.33 According to ANSI specifications which is the correct way of declaring main ( ) when it receives command line arguments?

(a)
```
main ( )
    {
        int argc; char*argv[ ];
    }
```

(b)
```
main (argc, argv)
    int argc; char*argv[ ];
```

(c) main (int argc, char argv [ ])

(d) None of the above

Q.34 What does the following 'C' program do?

```
main ( )
{
    unsigned int num;
    int i;
    scanf("%u", &num);
    for (i = 0; i < 16; i ++)
      printf("%d", (num<< i & 1 << 15)? 1 : 0);
}
```

(a) It prints binary equivalent of num

(b) It prints all odd bits from num

(c) It prints all even bits from num

(d) None of these

Q.35 What is the output of the following 'C' program?

```
main ( )
{
    unsigned int m = 32;
    printf("%x", ~m);
}
```

(a) ddfd

(b) ffdf

(c) fffff

(d) 0000

Q.36 What is the output of the following 'C' program?

```
main ( )
{
    extern int fun (float);
    int a;
    a = fun (3.14);
    printf("%d", a);
}
int fun (aa)
{
float aa;
{
    return ((int) aa);
} }
```

(a) Error
(b) 3.14
(c) 0
(d) 3

Q.37 Write a program to swap two numbers:

```
#include<stdio.h>
int main ()
{
    int a,b, temp;
    clrscr ();
    a=100, b=a+200;
    printf("before swapping a=%d and b=%d\n",a,b);
temp =a;
a=b;
b=temp;
printf ("After swapping a=%d and b=%d",a,b);
}
```

(a) a=200, b=100; a=100, b = 200
(b) a=100, b=300; a = 300, b = 100
(c) a=300, b=100; a = 100, b = 300
(d) a=200, b=200; a = 200, b = 200

Q.38 Consider a following declaration

int i =7;
float f = 8.5;

then which of the following statement is correct?

(i) (i+f)%4
(ii) ((int)(i+f))%2
(iii) ((float)(f+i))%4
(iv) ((int)f)%2
(a) Only (ii)
(b) (i),(iii)
(c) (ii),(iv)
(d) All of the above

Q.39 Trace the output:

```
        main ()
{
    float a =-4.9, b = 8.6; int c;
    c=floor (a)*a-(-b);
}
```

(a) 28.2
(b) 28
(c) 33.1
(d) 33

Q.40 Trace the final value of s and i

```
void main ( )
{   int c =1, s=0 i=1;
while (c<=5)
{
    s=s+i,
    i=i+2;
    c+=1;
}
print ("%d %d", s,i);
getch () ;
}
```

(a) 36 13
(b) 25 11
(c) 64 17
(d) 81 21

Q.41 The following program

```
main ()
{
    static int a [ ] ={7,8,9};
    printf ("%d",2[a]+a[2]);
}
```

(a) results in bus error

(b) results in segmentation violation error

(c) will not compile successfully

(d) none of the above

Q.42 Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main ()
{
int x, y, z;
x = y = z = 1;
z = ++x|| ++y && ++z;
printf ("x = %d y = %d z = %d\n", x, y, z);
return 0;
}
```

(a) x = 2 y = 1 z = 1

(b) x = 2 y = 2 z = 1

(c) x = 2 y = 2 z = 2

(d) x = 1 y = 2 z = 1

Q.43 Choose the correct statements.

(a) Constant expressions are evaluated at compile time

(b) String constants can be concatenated at compile time

(c) Size of array must be known at compile time

(d) None of the above

Q.44 In standard C, trigraph in the source program are translated

(a) before the lexical analysis

(b) after the syntax analysis

(c) before the recognition of escape characters in strings

(d) during the intermediate code generation phase

Q.45

```
#include <stdio.h>
void pri (int, int);
void printit (float, int);
int main( )
{
    float a = 3.14;
    int i = 99;
    pri (i, a);
    printit (a, i);
    return 0;
}
void pri (int i, int a)
{
    printf ("i=%d a=%f\n", i, a);
    printf ("a=%f i=%d\n", a, i);
}
void printit (float a, int i)
{
    printf ("a=%f i=%d\n", a, i);
    printf ("i=%d a=%f\n", i, a);
}
```

(a) i = 99 a = 3.000000
a = 3.000000 i = 99
a = 3.140000 i = 99
i = 99 a = 3.140000

(b) i = 99 a = 0.000000
a = 0.000000 i = 124503
a = 3.140000 i = 99
i = 99 a = 3.140000

(c) i = 99 a = 0
a = 0 i = 99
a = 3.14 i = 99
i = 99 a = 3.14

(d) None of the above

LEVEL–3

Q.46
```
#include <stdio.h>
int main( )
        {
        int i, j, k;
        for (j=1; j<=4; j++)
        {
            if (j*j==16)
                goto secretplace;
        }
        for (i=1; i<=5; i++)
        {
            k=i*i;
            j=k+2;
            secretplace:
                print("Murphy's second law\n");
                printf("Good computers are always priced...\n");
                printf("just beyond your budget\n");
        }
        return 0;
        }
```

select the correct statement

(a) Murphy's second law

Good Computers are always priced

.....

Just beyond your budget.

(b) Murphy's second law

Good Computers are always priced

.....

just beyond your budget.

After this the code causes a runtime exception.

(c) only runtime exception

(d) None of the above

GATE QUESTIONS

Q.47 Let x be an integer which can take a value of 0 or 1. The statement if (x = = 0) x = 1; else x = 0; is equivalent to which one of the following?

[IT-GATE 2004]
[1 Mark]

(a) x = 1 + x;

(b) x = 1 – x;

(c) x = x – 1;

(d) x = 1%x;

Q.48 A common property of logic programming languages and functional languages is

[GATE 2005]
[1-Mark]

(a) both are declarative

(b) all of the above

(c) both are procedural language

(d) both are based on λ-calculas

Q.49 What is the output printed by the following program?

```
# include <stdio.h>
int f(int n, int k) {
    if (n = = 0) return 0;
    else if (n%2) return f(n/2, 2*k) + k;
    else return f(n/2, 2*k) – k;
    int main () {
    printf("%d", f(20,1));
    return 0;
```

[IT-GATE 2005]
[2-Marks]

(a) 5

(b) 8

(c) 9

(d) 20

ANSWER KEY

1	d	2	b	3	c	4	a	5	a, c
6	a	7	a	8	d	9	a, b, c	10	a
11	a	12	a, c	13	d	14	a	15	b
16	c	17	c	18	a, b	19	a, b	20	all
21	a, b, c	22	b, c	23	all	24	d	25	d
26	b	27	a	28	b	29	c	30	c
31	d	32	c	33	c	34	a	35	b
36	d	37	b	38	c	39	d	40	b
41	d	42	a	43	a, b, c	44	a, c	45	b
46	b	47	b	48	a	49	c		

SOLUTIONS

S.3 (c)

We are defining **an array of 100 float value with typedef.**

S.4 (a)

Since we are doing/and * operations alternately, therefore **we may or may not yield the same value.**

S.5 (a, c)

rand( )		srand( )	
1.	**Type : int**	1.	**Type : void**
2.	returns a random positive integer	2.	use to initialize the random number generator

S.8 (d)

k = – 7. So, if 'k' is used as a Boolean variable, it will be treated as a true condition. So, !k will be false i.e., 0. So, 0 < ? !k is actually 0 < 0, which is false. **So, 0 will be printed.**

S.9 (a, b, c)

IF is not a keyword, because it is in upper.

S.10 (a)

Variable with integral values from 0 to n.

Then it's width is the integral part of $\log_2(n)+1$.

S.11 (a)

Since, we are declaring C as double, and we are printing C as character type.

∴ **Runtime error.**

S.12 (a, c)

In a 16-bit compiler like Turbo C/C++ the output will be 4 4 2. However, **in a 32-bit compiler like VC++ or gcc or Visualstudio, the compiler will report an error** since 32-bit compilers do not recognize near, far and huge pointers. In 32-bit compilers every pointer is 4 bytes wide.

S.14 (a)

A translation unit is **a set of source files seen by the compiler and translated as a unit-generally one '.c' file,** plus all header files mentioned in #include directives.

S.17 (c)

Output

i = 100

i = 1

i = 2

i = 2

i = 100

Explanation:

In the outermost block (a block is statements within a pair of braces) the variable i has been declared as an **auto** storage class variable, with an initial value 100. This value gets printed through the first **printf()**. Then the control reaches inside the next block, where again i is declared to have a value 1. This i is different , since it has been defined inside another block. The value of this i is then printed out through the second **printf()**. Inside the next block, this i is incremented to 2 and then printed out. Then the control reaches outside this block where another **printf()** is encountered. This **printf()** is within the same block in which the second i has been defined. Hence it prints out the value of this i, which is still 2. And now the control reaches outside the block in which the second i has been defined. Therefore the second i dies. The first i is however still active, since the control has not gone out of the block in which it has been defined. Hecne the last **printf()** prints out the value of this i, which is 100.

S.24 (d)

STEP 1: 0 ? 5 : 11, which is false, so, 11

STEP 2: 10 ? 11 : 12 which is true, so, **11**

S.25 (d)

All are required i.e., **Standard input, Standard output, Standard error.**

S.27 (a)

$(6)_{10}$ is represented as $(0110)_2$ in binary.

Similarly, $(10)_{10}$ is represented as $(1010)_2$

$$\begin{array}{rr} & 0110 \\ \& & 1010 \\ \hline & \mathbf{0010} \end{array} \Rightarrow 2$$

S.28 (b)

b = 10, c = 15

a = 11 + 15 = 26

$\therefore$ **a = 26, b = 11, c = 16**

S.29 (c)

The formula **p*(pow((i + 1), n))** is use for **calculating compound interest.**

S.30 (c)

f = 2 * ((8/5) + (4*(2))%(11))

= 2 * ((1 + 8)%(11))

= 2 * (9)

= **18 ⇒ data type ⇒ int**

S.31 (d)

The unary increment/decrement operator has the right to left associativity. Hence starting from right, **the first values is 3 and 4 going from right to left.**

S.32 (c)

f1, f2, f3

Hence function are evaluated according to the arithmatic expression precedence of arithmatic operator.

S.33 (c)

In ANSI standard declares the variables inside the parenthesis.

S.34 (a)

(num<< i & 1 <<15)? 1 : 0

This statement return either 0 or 1 depending on the condition inside the perenthesis. Hence printf becomes either printf **("%d",1);** or **printf("%d",0);.**

S.35 (b)

$m = (32)_{10} = (0000\ 0000\ 0010\ 0000)_2$

Now, $-m = (1111\ 1111\ 1101\ 1111)_2$

= **ffdf**

S.36 (d)

There is casting from float to int data type.

S.38 (c)

We cannot apply % operator on floating point value

(i) (i+f)%4 is invalid, because (i+f) is floating value

(iii) ((float)(f+i))%4 ⇒ floating value.

S.39 (b)

C = floor (a)*a–(–b)

= floor(–4.9)*(–4.9)+8.6

= –5*–4.9+8.6

= 33.1

But due to **int c ⇒ 33.**

S.40 (c)

In while loop, i < =5 Þ true

Then in th while loop condition is false.

1<=5	2<=5	3<=5	4<=5	5<=5
S=0+1=1	S=1+3=4	S=4+5 = 9	S=9+7=16	S=16+9=25
i=1+2+3	i=3+2=5	i=5+2=7	i=7+2=9	I=9+2=11
C=C+1=2	C=2+1=3	C=3+1=4	C=4+1=5	C=6

Q.41 (d)

a [2] will be converted to a(a+2).

*(a+2) can as well be written as *(2+a).

*(2+a) is nothing but 2[a]. So, a [2] is same as 2[a], which is same as *(2+a). So, it prints 9 + 9 = 18. Some of the modern compilers don't accept 2[a].

S.45 (b)

Output

i = 99 a = 0.000000

a = 0.000000 i = 124503

a = 3.140000 i = 99

i = 99 a = 3.140000

Explanation:

When **pri( )** is called the values passed to it are collected in the variable **i** and **a**. The first **printf( )** in **pri( )** prints out the value of **i** correctly, but value of **a** gets messed up because we are trying to print an integer value collected in a using the specification **%f**. The second **printf( )** messes up both the values since the first specification itself in this **printf( )** is wrong. Remember that once the output of one variable goes awry, the **printf( )** messes up the output of the rest of the variables too.

When **printit( )** is called **a** has been declared as **float** whereas **i** has been declared as an **int**. Therefore both the **printf( )**s output the values as expected.

S.46 (b)

Murphy's second law

Good computers are always priced...

just beyond your budget

After this the code causes a runtime exception

Explanation:

Look at the first **for** loop. The moment **j*j** equals 16, the **goto** statement is executed, which takes the control to **secretplace** incide the second **for** loop. This is perfectly acceptable, **goto** can virtually take the control anywhere– even deep inside a **for** loop. Having reached the **secretplace,** Murphy's second law is printed out and then the control reaches the closing brace of the **for** loop. As a result, control jumps to the beginning of loop i.e. to **i++**. Here it attempts to increment **i**. However, since **i** has not been initialized it would hold a garbage value because we entered the loop directly at **secretplace** as a result of which **i = 1** couldn't get executed. That is why a runtime exception is raised with a message. 'The variable 'i' is being used without being initialized.

S.47 (b)

Given, C code is:

```
if (x = = 0)
x = 1;
else
x = 0;
```

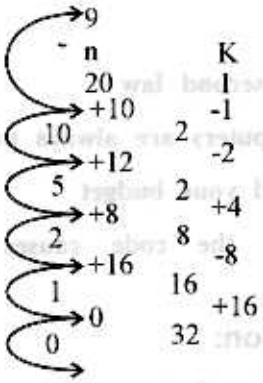
If x is zero then set x = 1 else set x which is same as **x = 1 – x.**

S.48 (a)

A common property of logic programming language and functional languages is **both are declarative because we declare any statement before we will use it.**

S.49 (c)

The program execution sequence is



So final result is 9.

4.2 CONTROL STATEMENTS

LEVEL–1

Q.1 Which of the following is possible with the help of "goto" statement?

(i) Branching around statements or group of statements under certain conditions.

(ii) Jumping to the end of a loop under certain conditions, thus by passing the remainder of the loop during the current pass

(iii) Jumping completely out of a loop under certain conditions, thus determining the execution of a loop.

(a) (i), (ii), (iii)

(b) (i), (iii)

(c) only (i)

(d) (ii), (iii)

Q.2 Consider for loop in a C program. If the condition is missing

(a) It is assumed to be present and taken to be true

(b) It result in a syntax error

(c) It is assumed to be present and taken to the false

(d) execution will be terminated abruptly

Q.3 In the following 'C' program, find out the error in the 'while' loop, if any?

```
main ( )
{
    int i = 1;
    while ( )
    {
        printf("%d", i++);
        if (i > 10)
        break;
    }
}
```

(a) There should be at least a semicolon in the while ( )

(b) The while loop should be replaced by for loop

(c) No error

(d) The condition in the while loop is a must

Q.4 A "switch" statement is used to

(a) switch between functions in a program

(b) to choose from multiple possibilities which may arise due to different values of a single variable

(c) switch from one variable to another variable

(d) to use switching variable

Q.5 Static variables are sometimes called

(a) class variables

(b) functional variables

(c) dynamic variables

(d) auto variables

Q.6 Trace the o/p

```
main ()
{   int x =10,y=10,z=5 i;
    i=x<y<z;
    pirntf("/n%d",i);
}
```

(a) 1

(b) 0

(c) Error

(d) 5

Q.7 Consider the following program fragment

```
sum = 0;
do  {   scanf("%d",&x)
        if (x>0) printf ("%d",x);
        else
        if (x==0) break;
        sum +=x;
    }   while (sum < 10);
```

What would be the output if input is 5,2,0,3,0?

(a) 5 2

(b) 5 2 0

(c) 5 2 0 3

(d) 5 2 3

Q.8 The following program fragment:

```
for (i=1; i<5 ;i+=3), { }
print ("%d",i);
```

results in

(a) a syntax error

(b) an execution error

(c) printing of 7

(d) printing of 16

Q.9 What is the output of the following program?

```
main()
{
int x,y,z;
x= 2;
y=1;
z=1;
if (x>y+z)
    printf("Hello!n");
else if (x<y+z)
printf(Hi!/n");
else
printf("Hey!/n");
}
```

(a) Hello!

(b) Hi!

(c) Hey!

(d) none of these

Q.10 Which of the following statements are correct about the C program given below?

```
#include<stdio.h>
int main()
{
    int x = 10, y = 100% 90;
    for (i = 1; i <= 10; i++);
    if (x != y);
        printf ("x = %d y = %d\n", x,y);
    return 0;
}
```

(a) The printf() function is called 10 times.

(b) The program will produce the output x = 10 y = 10.

(c) The ; after the if (x! = y) will NOT produce an error.

(d) The program will not produce any output.

(e) The printf() function is called infinite times.

Q.11 Point out the error, if any, in the following program.

Find out put :

```
#include < stdio.h>
int main ()
{
    int i = 1;
    switch (i)
{
    case 1 * 2 + 4:
    printf("\n Bottle per rent-inquired within.\n");
    break,
    case 1 :
    printf("\n Radioactive cats have 18 half-lines.\n");
    break;
}
```

(a) Bottle for rent-inquire within

(b) Radioactive cats have 18 half-lives.

(c) error

(d) 6

Q.12 Rewriting following of statements using conditional operators will result into:

```
int a = 1, b;
if (a > 10)
    b = 20;
```

(a) `int a = 1, b;`
`a > 10? b = 20 : ;;`

(b) `int a = 1, b;`
`a > 10? b = 20 : (0);`

(c) `int a = 1, b;`
`a > 10? (b = 20) : ,;`

(d) `int a = 1, b, dummy;`
`a > 10? b = 20: dummy = 1;`

Q.13 Choose the statements that are syntactically correct.

(a) `/* Is /* this a valid */ comment * /`

(b) `for (; ;) ;`

(c) `return ;`

(d) `return (5 + 2);`

Q.14 Consider the statements

```
putchar(getchar());
putchar(getchar());
```

If

a

b

is the input, the output will be

(a) an error message

(b) this can't be the input

(c) ab

(d) a b

Q.15 Integer division results in

(a) truncation

(b) rounding

(c) overflow

(d) None of the above

Q.16 Pick the operators that associate from the right.

(a) ?:

(b) +=

(c) =

(d) <

Q.17 The following code fragment

```
int x, y = 2, z, a;
x = (y *= 2) + (z = a = Y);
print f("%d", x);
```

(a) prints 8

(b) prints 6

(c) prints 6 or 8 depending on the compiler implementation

(d) is syntactically wrong

Q.18 The following statement

```
print f ("%f", 9/5);
```

prints

(a) 1.8

(b) 1.0

(c) 2.0

(d) none of the above

Q.19
```
#include <stdio.h>
int main()
{
    unsigned int ch = 0;
    for (ch = 65; ch <= 225;)
    printf ("%d %c\n", ch, ch++);
return 0;
}
```
(a)
```
65 A
66 B
... ..
... ..
125 }
126 ~
... ..
... ..
255
... ..
... ..
65 A
66 B
... ..
... ..
```
(b)
```
66 A
67 B
... ..
... ..
125 |
126 }
... ..
... ..
255
256
```
(c)
```
65 A
66 B
... ..
... ..
125 }
126 ~
... ..
... ..
255
256
```
(d)
```
66 A
67 B
... ..
... .
125 |
126 }
... ..
... ..
255
... ..
... ..
66 A
67 B
... ..
... ..
```

Q.20 Which of the following comments about **for** loop are correct?

(a) Using 'break' is equivalent to using a 'goto' that jumps to the statement immediately following the loop.

(b) Continue is used to by-pass the remainder of the current pass of the loop.

(c) If comma operator is used, then the value returned is the value of the right operand.

(d) If can always be replaced by a 'while' loop.

Q.21 The 'switch' feature

(a) can always be replaced by a nested if-then-else clause

(b) enhances logical clarity

(c) can't always be replaced by a nested if-then-else clause

(d) none of the above

Q.22 Choose the best answer

Storage class defines

(a) the data type

(b) the scope

(c) the scope and permanence

(d) the scope, permanence and data type

Q.23 Consider the following program segment.

```
i = 6720 ; j = 4 ;
while ((i%j) = = 0)
{ i = i / j ;
  j = j + 1;
}
```

On termination j will be the value

(a) 4

(b) 8

(c) 9

(d) 6720

Q.24 The program

```
main ()
{
    int i = 5;
    i = (++ i) / (i ++);
    print f("%d", i);
}
```

prints

(a) 2

(b) 5

(c) 1

(d) 6

Q.25 The process of transforming one bit pattern into another by bit-wise operations is called

(a) masking

(b) pruning

(c) biting

(d) chopping

Q.26 The operation of a staircase switch best explain the

(a) OR operation

(b) AND operation

(c) XNOR operation

(d) XOR operation

Q.27 The number of possible values of m, such that m & 0x3f equals 0x23 is

(a) 1

(b) 2

(c) 3

(d) 4

LEVEL–2

Q.28 If a = 9, b = 5 and c = 3, then the expression (a -- a/b * b%c) > a%b%c evaluates to

(a) true

(b) false

(c) invalid

(d) 0

Q.29 In following code, find out the error in 'while' loop, if any

```
main ( )
{
  int i = 1;
  while ( )
        { printf("%d", i++);
          if (i > 10)
          break;
        }
}
```

(a) The while loop should be replaced by for loop

(b) There should be at least a semicolon in the while

(c) The condition in the while loop is a must

(d) No error

Q.30 Consider a following declaration

```
main ()
{ sum = 0;
  do { scanf("%d", &i);
       if (i < 0)
       { i = –1;
         ++ flag;
       } sum + = i;
    } while (i! = 0);
}
```

which of the following statement is correct?

(i) The program segment itself is a compound statement

(ii) The do-while statement, which is embedded in the program segment, contains a compound statement.

(iii) The if statement, which is embedded in the do-while statement, contains a compound statement.

(a) only (ii) & (iii)

(b) (i), (ii), (iii)

(c) only (i) & (ii)

(d) neither of (i), (ii), (iii)

Q.31 Consider the following declaration for (x = 1; x <=10; x++)

```
{
  for (y = x; y <= 10; y++)
  printf ("%d", x);
  printf ("\n");
}
```

The (sum of all elements of 4th row) – (sum of all elements of 7th row) = _____

(a) 7

(b) 0

(c) 5

(d) 3

Q.32 Consider the following declarations:

```
count = 0;
do
{
  printf("%d", text [count]);
        ++count;
}
  while (text [count]! = '*');
```

Which of the following statement is incorrect?

(a) If we enter '*' as first character then also this program will execute atleast once.

(b) For the above code, if we replace do–while by while then program execution is slightly faster.

(c) We can replace text [count] in output statement as text [count ++], without writing afterwards.

(d) (a), (b) & (c) are wrong.

Common Data For Questions 33 & 34:

```
main ( )
{       int n = 1, sum = 0;
        while (n <= 10)
        {
                sum + = n * n ++;
        }
        printf("sum = %d\n", sum),
}
```

Q.33 What is the output of the above code?

(a) 2121

(b) 385

(c) 440

(d) Indefinite loop

Q.34 If the statement 4 is replaced by

Sum + = n ++ * n ++;

What will be the difference between the new and old output?

(a) 0

(b) 218

(c) 220

(d) –218

Q.35

```
void main ( )
{
    int stud [ ] [2] = {1234, 56, 1212, 33, 1434,
80, 1312, 778, 1203, 75};
    // let stud [ ] [2] start at 4001.
    int i, j, sum = 0, j=0;
    for (i = 0; i <= 4; i ++, j ++)
    {
            if ( j = = 2) j = 0;
            sum + = *(*(stud + i) + j),
    }
            printf("%d", sum),
}
```

(a) 2735

(b) 6717

(c) 3880

(d) 4682

Q.36 What is the output of the following 'C' fragment?

```
for (i = 1, j = 10; i < 6; ++i, – –j)
printf("%d%d", i, j);
```

(a) 1 1 1 1 1 9 9 9 9 9

(b) none of these

(c) 1 10 2 9 3 8 4 7 5 6

(d) 1 2 3 4 5 10 9 8 7 6

Q.37 Match the following:

Group I		Group II	
1.	Type declaration instructions	a.	k = i*234+n-7;
2.	Input/Output instructions	b.	for(i=0;i<=10;i++);
3.	Arithmetic instructions	c.	gets( );
4.	Control instruction	d.	char name, code;

(a) 1 – b, 2 – a, 3 – d, 4 – c

(b) 1 – d, 2 – c, 3 – a, 4 – b

(c) 1 – c, 2 – b, 3 – a, 4 – d

(d) 1 – a, 2 – d, 3 – b, 4 – c

Q.38 How many times will the printf statement be executed?

```
main ( )
{
int n;
n = 10;
while (n < 10) {
                    printf("hello");
                    – – n;
}       }
```

(a) once

(b) 9

(c) 10

(d) never

Q.39 Consider the program fragment

```
switch(choice){
case 'R' : printf("RED");
case 'W' : printf("WHITE");
case 'B' : printf("BLUE");
default : printf("ERROR");
   break;
          }
```

What would be the output if choice = 'R'?

(a) RED WHITE BLUE

(b) RED

(c) RED ERROR

(d) RED WHITE BLUE ERROR

Q.40 Following program returns

```
void summation (n)
int n;
{
int i; sum = 0;
i = 1;
for (i = 1; i <= n, i++)
    sum + = i;
}
```

(a) sum of 1, 2, ..., n

(b) nth number

(c) sum of the n number

(d) none of these

Q.41 Consider the following program

```
main ( )
{
int x = 2, y = 5;
if (x < y) return (x = x + y) ;
else printf("z1") ;
printf("z2") ;
}
```

Then

(a) this will result in compilation error

(b) output is z2

(c) output is z1z2

(d) none of these

Q.42 The following program fragment:

```
int k = –7,
printf ("%d",0<!k);
```

(a) prints0

(b) prints a non zero vlaue

(c) is illegal

(d) prints an unpredicateble value.

Q.43
```
# include<stdio.h>
#include<conio.h>
voidmain()
{
int i;
clrscr();
i=1;
while (i<=5)
{
    print("%d",i);
    i++;
}
}
```
(a) 11145
(b) 12145
(c) 12345
(d) none of these

Q.44
```
# include<stdio.h>
#include<conio.h>
void main()
{
    int i ;
    clrscr ()
    i=1;
    while (i<=5)
    {
        printf("%d",i);
        if (i==3)
        con·inue;
        i++;
        }
}
```
(a) 113...
(b) 123...
(c) 133...
(d) none of these

Q.45 Find out the error in the following program:
```
main ()
{
    int mark;
    char grade;
    switch (mark)
    {
        case 5: grade = 'A'; break;
        case4: grade = 'B'; break;
        case4; grade = 'B' break;
        defualt: grade = 'C'; break;
    }*/switch*/
}
```
(a) switch statement cannot have more than three lables
(b) case lables cannot be numbers
(c) no two lables may be identical
(d) none of the above

Q.46 Consider the foloowing declaration:
```
switch (color)
{
    case 'r' :
    case 'R' : printf("RED");
               break;
    case 'g' :
    case 'G' : printf("GREEN");
               break;
default: printf ("BLACK");
               break;
}
```
Which of the following statement is correct.
(a) The above declaration is incorrect due to first case value
(b) The above declaration is incorrect due to first and third case values
(c) The above declaration perform only default condition whether intput is r, R, g, G
(d) The above declaration works properly for g, r, R, G and any other intput value also.

Q.47 The following statement:

```
if (a>b)
if(c>b)
printf("one");
else
if(c==a) printf("two");
else printf("there");
else printf("four");
```

(a) results in a syntax error

(b) prints four if c<=b

(c) prints two if c<=b

(d) prints four if a <=b.

Q.48 What is the value of "average" after the following program is executed?

```
main ( )
{
int sum, index;
index = 0;
sum = 0;
for (;;) {
sum = sum + index;
++ index;
if (sum >= 100) break;
    }
average = sum/index;
```

(a) 91/14

(b) 105/14

(c) 91/13

(d) 105/15

Q.49 What will be the output of the following program?

```
{
int i = 1234; j = 0177, k = 0xa08c;
printf("%8d%8o%8x/n", i, j, k),
}
```

(a) 12340777ao80

(b) 00001234 01aa ax

(c) 1234 1 abc a08c

(d) 1234177 a08c

Q.50 The O/P of following program will be.

```
#include <stdio.h>
int main()
{
    int a = 10, b;
    a >= 5? b = 100 : b = 200;
    printf ("%d\n", b);
    return 0;
}
```

(a) 100

(b) 200

(c) error

(d) compile error

Q.51 Which of the following is the correct output for the program given below?

```
#include <stdio h>
int main ()
{
    char j = 1;
    while (j <= 255)
    {
        printf ("%d", j);
        j = j + 1;
    }
    printf ("\n");
    return 0;
}
```

(a) 1 2 3 .. 127

(b) 1 2 3 .. 255

(c) 1 2 3 ... 254 255 0 1 2 3 ... 254 255... infinite times

(d) 1 2 3 ... 127 128 0 1 2 3 ... 127 128 ... infinite times

(e) 1 2 3 ... 127 -128 -127 -126 ... -2 -1 0 1 2 ... 127 -128 -127... infinite times

Q.52 What will be the output of the following program?

```
#include <stdio.h>
{
    char ch;
    ch = 'A';
    printf ("The letter is");
    printf ("%c", ch >= 'A' && ch <= 'Z' ? ch + 'a' - 'A' : ch);
    printf ("nNow the letter is");
    printf ("%c\n", ch>='A' && ch<='Z' ? ch : ch + 'a' - 'A');
    return 0;
}
```

(a) The letter is
Now the letter is

(b) The letter is a
Now the letter is A.

(c) Error

(d) No output

Q.53 Which of the following statements are correct about the program given below?

```
#include <stdio.h>
int main()
{
    unsigned int num;
    int i ;
    scanf ("%u", &num);
    for (i = 0; i < 16; i++)
        printf ("%d", (num<<i & 1<<15)? 1 : 0);
        printf ("\n");
        return 0;
}
```

(a) It prints all even bits from num.

(b) It prints all odd bits from num.

(c) It prints binary equivalent of num.

(d) None of the above

Q.54

```
#include <stdio.h>
int function (int, int);
int main ()
{
    int i = 135, a = 135, k;
    k = function (!++i, !a++);
    printf ("i=%d a=%d k=%d\n", i, a, k);
    return 0;
}
int function (int j, int b)
{
    int c;
    c = j + b;
    return (c);
}
```

(a) i = 136 a = 135 k = 271

(b) i = 136 a = 136 k = 0

(c) i = 136 a = 136 k = 272

(d) None of the above

Q.55

```
#include <stdio.h>
#define ISUPPER(x) (x>=65 && x<=90)
#define ISLOWER(x) (x>=97 && x<=122)
#define ISALPHA(x) (ISUPPER(x) || ISLOWER(x))
int main ( )
( char ch='+';
if (ISALPHA(ch))
    printf("ch contains an alphabet\n");
else
    printf("ch doesn't contain an alphabet\n");
return 0;
}
```

(a) ch contains an alphabet

(b) ch doesn't contain an alphabet

(c) error

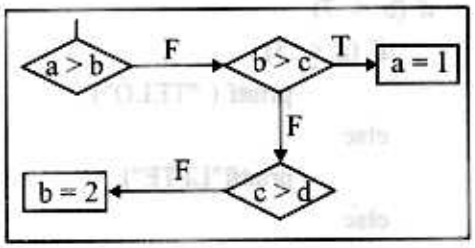
(d) no output

Q.56 #include <stdio.h>
int main ()
{
char s[] = "C it yourself";
int i = 0;
while (s[i])
{
if (s[i]! = '')
s[i] = s[i] + 1;
i++;
}
printf ("%s\n", s);
return 0;
}
(a) C it yourself
(b) B hs xntqrdke
(c) Error
(d) D ju zpvstfmg

Q.57 The following program fragment
int i = 5 ;
do {putchar (i + 100); print f("%d", i – –);}
while (i) ;
results in the printing of
(a) i5h4g3f2el
(b) i4h3g2fle0
(c) an error message
(d) None of the above

Q.58 What will be value of count after the following program is executed ?
main ()
{
int count, digit=0;
count=1;
while(digit<=9)
{
printf("%d", count);
++ digit;
}
(a) 1001011101
(b) 1111111011
(c) 0111111111
(d) 1111111111

Q.59 Consider the following flow chart.



Which of the following, correctly implements the above flow chart?
(a) if (a > b)
if (b > c)
a = 1;
else if (c > d)
b = 2;
(b) if (a <= b)
if (b > c)
a = 1;
else if (c <= d)
b = 2;
(c) if (a > b);
else if (b > c)
a = 1;
else if (c <= d)
b = 2;
(d) if (a > b);
else if (b > c)
a = 1;
else if (c > d),
else b = 2;

Q.60 Consider the program fragment
j = 2;
while ((i % j) ! = 0)
j = j + 1;
if (j < i) printf ("%d", j);
If i > = 2, then the value of j, will be printed only if
(a) i is prime
(b) j does not divide i
(c) j is odd
(d) i is not prime

Q.61 The following program
main()
{

```
float a = .5, b = .7;
if (b < .7)
    if (a < .5)
        printf ( "TELO");
    else
        printf("LTTE");
else
        printf ("JKLF");
}
```

outputs

(a) LTTE

(b) TELO

(c) JKLF

(d) PLO

Q.62 Consider the following program.

```
main ( )
{ putchar ( 'M') ;
first ( );
putchar ( 'm') ; }
first ( )
{_____}
second ( )
{ putchar ( 'd'); }
```

If Madam is the required output, then the body of first ( ) must be

(a) empty

(b) second ( ); putchar ('a') ;

(c) putchar ('a'); second (); printf ("%c", 'a');

(d) None of the above

Q.63

```
#include <stdio.h>
#define a 10
main ( )
{ #define a 50
    print f ('%d' , a) ;
}
```

The output will be :

(a) 10

(b) Error

(c) 50

(d) Redec location # define not allowed within main.

LEVEL–3

Q.64 Trace the O/P

```
sum = 0;
for (i=1;i<=10;i++)
if (i%2==0) sum+=i;
printf("%d" sum);
```

(a) 40

(b) 45

(c) 0

(d) 30

Q.65 Consider the following declaration

(i)
```
for (j = 2; j <= 13; ++j)
{
  sum = 0;
  i = 2;
  while ( i < 100)
  { sum + = i;
    i + = j;
  }
  printf("%d", sum);
}
```

(ii)
```
for (j = 2; j <=13; ++j)
        {
          sum = 0;
          i = 2;
do
          {  sum + = i;
             i + = j;
          }  while (i < 100);
          printf("%d", sum);
        }
```

(a) The (A) and (B) both finds the series 2 + 4 + 6 + ... + 98

(b) The (A) is find the sum of series 2 + 4 + 6 + ... + 98 but (B) is not logically correct

(c) The (A) is find the sum of series 2 + 4 + 6 + ... + 98 and (B) is find the sum of series 2 + 4 + 8 +..... + 98

(d) The (A) and (B) performs the same operation of finding the sum of series 2 + 4 +8 + ... + 98

Q.66 Trace the output

```
main ()
{ int i = 0, x =0;
do {
    if (i%5==0)
    { x ++;
print f ("%d",x);
} ++i;
} while (i<20);
print ("%d",x)
}
```

(a) 1 2 3 4 5

(b) 1 2 3 5 7

(c) 1 2 3 4 4

(d) None of these

Q.67 Trace the output

```
main ( )
{
    int i, j, k, x = 0;
    for (i = 0; i < 5; ++i)
{
    for (j = 0; j < i; ++j)
    {
        k = (i + j – 1);
        if (k%2 = = 0)
        x + = k;
        else
        if (k%3 = = 0)
        x + = k + 2;
        printf("%d", x);
    }
    printf("%d", x); }
}
```

(a) 0 0 2 4 5 9 10 14 14 20 20

(b) 0 0 2 4 5 9 10 13 13 19 19

(c) 0 0 2 4 5 7 9 11 13 15 15

(d) 0 0 2 4 6 8 10 12 14 16 18

(e) 0 0 2 2 4 9 13 18 22 22 28 28

Q.68 Trace the output:

```
main ( )
{
    int i, j, k, x = 0;
    for (i = 0; i < 5; ++i)
    for (j = 0; j < i; ++j)
    {
        switch ( i + j – 1)
            {
                case – 1 :
                case 0 : x + = 1;
                        break;
                case 1 :
                case 2 :
                case 3 : x + = 2;
                default : x + = 3;
            }
            printf("%d", x);
        }
        printf("%d", x);
}
```

(a) 1 4 7 10 13 16 19 22 25 28 28

(b) 1 6 11 16 21 24 29 32 35 38 41

(c) 1 6 11 16 21 24 29 32 35 38 38

(d) 1 4 7 12 15 20 23 28 31 36 41

Q.69 Consider the following program fragement:

```
if (a>b)printf("a>b");
else printf("else part");
printf("a<=b");
```

a<=b will be printed if

(a) a>b

(b) a<b

(c) a==b

(d) none of the above.

Q.70 Consider the following declaration

```
main ( )
{   int n, r, x = 0, a,
    do
        {
           scanf("%d", &n);
        }
        while (n <= 0);
        a = n ;
        for ( ; a > 0;)
          { r = a % 10;
            x = x * 10 + r;
            a = a/10;
          }
          if (x = = n)
                :
```

The above code can be used for _____

(a) finding palindrome number
(b) finding the reverse of the number
(c) finding Armstrong number
(d) we cannot use due to errors

Q.71

```
void main ( )
{   int n, x, y;
    printf("Enter n");
    scanf("%d", &n);
    for (x = 1; x <= n; x ++)
    {
        for (y = 1; y <= x; y++)
        printf("%d", y);
        printf("\n");
        }
}
```

Due to above declaration _____

(a) Due to stack overflow this program did not work.
(b) Starting from 1 to n, a each line contains 1 to digits which is equals to the number of line i.e. 1 is printed on first line, 1 2 is printed on second line and so on.
(c) Starting from 1 to n digits are printed on one line and such a n lines are printed.
(d) Starting from 1 to n digits are printed on n different lines.

Q.72

```
# include <stdio.h>
#include<conio.h>
void main( )
{
    int i;
    clrscr( );
    for(i=1;i<=5;i++)
    {
    if(i==3)
    break;
    printf("%d",i);
    }
}
```

(a) 1245
(b) 12345
(c) 1234
(d) 12

Q.73

```
# include <stdio.h>
#include<conio.h>
void main()
{
    int i;
    clrscr();
    for (i=1 ;i<=5;i++)
    {
    if(i==3)
    continue ;
    printf("%d",i);
    }
}
```

(a) 1145
(b) 1245
(c) 1345
(d) 12345

Q.74 Consider the following declaration:

```
void main ( )
{
    int n, x, y, s = 40;
    scanf("%d", &n);
    for (x = 1, x <= 2 * n – 1; x + = 2)
    {
        for (y = 1; y < = s; y++)
        printf(" ");
        for (y = 1; y <= x; y++)
        printf("%d", y);
        for (y = x – 1; y >=1; y—)
        printf("%d", y);
        printf("\n");
        s = s – 4;
    }
}
```

The sum of all elements in the second row, which is printed by above code is (take n>1)

(a) 7

(b) 9

(c) 18

(d) 4

Q.75

```
# include<stdio.h>
#include<conio.h>
void main( )
{
    int i;
    clrscr( );
    for(i=1;i<=5;i++)
    {
    if (i==3)
    goto end;
    printf("%d",i);
    end;
    }
}
```

(a) 12345

(b) 1234

(c) 1245

(d) 123

Q.76 Consider a following declaration

```
main ( )
{  int i = 2; n = 3;
   if (i < 5)
    {
      for ( ; n ≤ 4; n ++)
      {
         printf("%d", n);
      }
    }
}
```

considering the complexity for the condition is O(2)

What will be the complexity for above code

(a) O(4)

(b) O(2)

(c) O(6)

(d) O(5)

Q.77

```
#include <stdio.h>
#define P(format, var) printf("var=%format\n", var)
int main ()
{
    int i = 3;
    float a = 3.14;
    P(d, i);
    P(f, a);
    return 0;
}
```

(a) Unexpected end of file in conditional error.

(b) var = 0.000000 format
var = 3.140000 format

(c) var = 0.000000 ormat
var = 3.140000 ormat

(d) No output

Q.78 The following program fragment:

```
int i=5;
do {
    putchar (i+100);
    printf("%d",i- -);
    }
while (i);
```

results in the printing of

(a) i5h4g3f2el

(b) i4h3g2fle0

(c) an error message

(d) none of the above

Common Data For Question 79 & 80:

Consider the following program fragment:

```
d = 0;
for (i = 1; i < 31; ++i)
    for (j = 1; j < 31; ++j)
        for(k = 1; k < 31; ++k)
          if(((i + j + k) % 3) = = 0)
d=d+1;
printf("%d", d) ;
```

Q.79 The output will be

(a) 9000

(b) 27000

(c) 3000

(d) none of the above

Q.80 The number of additions performed by the above program fragment is

(a) 27000

(b) 27000 × 3

(c) 9000 + 3 × 27000

(d) 9930 + 27000 × 3

GATE QUESTIONS

Q.81 An unrestricted use of the "go to" statement is harmful because of which of the following reason (s): **[GATE 1989]**

(a) It makes it more difficult to verify programs.

(b) It makes program more inefficient.

(c) It makes it more difficult to modify existing programs.

(d) It results in the compiler generating longer machine code.

Q.82 Given the programming constructs (i) assignment (ii) for loops where the loop parameter cannot be changed within the loop (iii) If then else (iv) forwards goto (v) arbitrary goto (vi) non recursive procedure call (vii) recursive produre/ function call (viii) repeat loop, which constructs will you not include in a programmin language such that it should be possible to program the terminates (i.e. halting) function in the same programming language. **[GATE 1999] [2-Marks]**

(a) (ii),(iii),(iv)

(b) (v),(vii),(viii)

(c) (vi),(vii),(viii)

(d) (iii),(vii),(viii)

Q.83 Consider the following C function definition

```
int Trial (int a, int b, int c)
{
    if ((a >=b) && (c<b)) return b;
    else if (a>=b) return Trial (a,c,b)
    else return Trial (b,a,c)
}
```

The function Trial: **[Gata 1999] [2-Marks]**

(a) finds the maximum of a,b and c

(b) finds the minimum of a,b and c

(c) finds the middle number of a,b,c

(d) none fo the above

Q.84 In the following C program fragment j, k n and TwoLog_n are integer variables, and A is an array of integers. The variable n is intialized to an integer ≥ 3, and TwoLog_n is initialized to the value of $2^{*}\lceil \log_2(n) \rceil$

```
for (k = 3; k <= n; k++)
    A[k] = 0;
for (k = 2; k <= TwoLog_n; k ++)
    for (j = k + 1; j <= n; j ++)
        A[j] = A[j] || (j%k);
for (j = 3; j <= n; j ++)
    if (!A[j]) printf("%d", j);
```

The set of numbers printed by this program framgment is **[GATE 2003] [2-Marks]**

(a) { }

(b) {m | m ≤ n, (∃i) [m = i!]}

(c) {m | m ≤ n, (∃i) [$m = i^2$]}

(d) {m | m ≤ n, m is prime}

Q.85 What does the following algorithm approximate? (Assume $m > 1, \epsilon > 0$).

```
x = m ;
y = 1 ;
while (x – y > ∈)
                {    x = (x + y)/2;
                     y = m/x;
                }
print(x);
```

[GATE 2004] [2-Marks]

(a) $m^{1/3}$
(b) $m^{1/2}$
(c) log m
(d) m^2

Q.86 Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let $n = d_1 d_2 \ldots d_m$

```
int n, rev ;
rev = 0 ;
while ( n > 0) {
        rev = rev * 10 + n % 10 ;
        n = n / 10 ;
}
```

The loop invariant condition at the end of the i^{th} iteration is **[GATE 2004] [2-Marks]**

(a) $n = d_1\ d_2 \ldots d_m$ or $rev = d_m \ldots d_2\ d_1$
(b) $n \neq rev$
(c) $n = d_{m-i+1} \ldots d_{m-1}\ d_m$ or $rev = d_{m-i} \ldots d_2\ d_1$
(d) $n = d_1\ d_2 \ldots d_{m-i}$ and $rev = d_m\ d_{m-1} \ldots d_{m-i+1}$

Q.87 Consider the following C function

```
int     f(int n)
{ static int i = 1 ;
        if (n >= 5) return n ;
        n = n + i ;
        i ++;
        return f(n) ,
}
```

The value of returned by f(1) is **[GATE 2004] [2-Marks]**

(a) 6
(b) 8
(c) 7
(d) 5

Q.88 Consider the following C program

```
main ( )
{   int x, y, m, n;
    scanf("%d%d", &x, &y);
    / * Assume x > 0 and y > 0* /
    m = x;          n = y;
    while (m! = n)
          {     if (m > n)
                m = m – n;
                else
                n = n – m;
          }
    printf("%d", n);
    }
```

The program computes **[GATE 2004] [2-Marks]**

(a) the least common multiple of x and y
(b) the greatest common divisor of x and y
(c) x ÷ y, using repeated subtaction
(d) x mod y using repeated subtraction

Q.89 What is the output of the following program?

```
#include <stdio.h>
int funcf(int x);
int funcg(int y);
main ( )
{
    int x = 5, y = 10, count;
    for (count = 1; count <=2; ++count){
                        y+=funcf(x) + funcg(x);
                        printf("%d", y);
    }
}
funcf(int x){
        int y;
        y = funcg(x);
        return(y);
}
```

```
funcg(int x){
        static int y = 10;
        y+=1;
        return(y + x);
}
```

[IT-GATE 2004]

[2-Marks]

(a) 43 80

(b) 42 74

(c) 33 37

(d) 32 32

Q.90 Consider the following C-program

```
void foo (int n, int sum) {
        int k = 0, j = 0;
if (n = = 0) return;
k = n % 10; j = n/10;
        sum = sum + k;
        foo (j, sum);
printf("%d", k);
        }
        int main ( ) {
int a = 2048, sum = 0;
        foo (a, sum);
printf("%d/n", sum);
        }
```

What does the above program print?

[GATE 2005]

[2-Marks]

(a) 2, 0, 4, 8, 0

(b) 8, 4, 0, 2, 0

(c) 2, 0, 4, 8, 14

(d) 8, 4, 0, 2, 14

Q.91 Consider the following C-function in which a[n] and b[m] are two sorted integer arrays and c[n + m] be another integer array.

```
void xyz (int a[ ], int b[ ], int c[ ]) {
    int i, j, k;
    i = j = k = 0;
    while ((i < n))&&(j < m)
        if (a[i] < b[j] c[k++] = a[i++];
        else c[k++] = b[j++];
    }
```

Which of the following condition(s) hold(s) after the termination of the while loop?

[GATE 2006]

[2-Marks]

(i) $j < m$, $k = n + j - 1$, and $a[n - 1] < b[j]$ if $i = n$

(ii) $i < n$, $k = m + i - 1$, and $b[m - 1] \leq a[i]$ if $j = m$

(a) neither (i) nor (ii)

(b) either (i) or (ii) but not both

(c) only (i)

(d) only (ii)

Q.92 Consider the following segment of C-code

```
int j, n;
    j = 1;
while (j <= n)
    j = j*2;
```

The number of comparisons made in the execution of the loop for any $n > 0$ is

[GATE 2007]

[1-Mark]

(a) $\lfloor \log_2 n \rfloor + 1$

(b) n

(c) $\lceil \log_2 n \rceil$

(d) $\lceil \log_2 n \rceil + 1$

ANSWER KEY

1	a	2	b	3	d	4	b	5	a
6	a	7	a	8	c	9	c	10	b, c
11	b	12	d	13	b, c, d	14	b	15	a
16	a, b, c	17	c	18	d	19	b	20	all
21	a, b	22	c	23	c	24	a	25	a
26	d	27	d	28	a	29	c	30	b
31	b	32	d	33	b	34	c	35	d
36	c	37	b	38	d	39	d	40	a
41	d	42	a	43	c	44	b	45	c
46	d	47	d	48	d	49	d	50	b
51	e	52	b	53	c	54	b	55	b
56	d	57	a	58	d	59	b, c, d	60	d
61	a	62	c	63	c	64	d	65	a
66	c	67	e	68	c	69	a, b, c	70	a
71	b	72	d	73	b	74	b	75	c
76	b	77	c	78	b	79	a	80	d
81	a, b	82	b	83	c	84	a	85	b
86	d	87	c	88	b	89	a	90	a
91	b	92	a						

SOLUTIONS

S.1 (a)

We can perform (i), (ii), (iii) with the help of "goto" statment which is actually an unconditional branching statement.

S.6 (a)

x = 10, y = 10, z=5

$\Rightarrow$ (x<y)<z = 0 < z = True

As it is left to rigth associativity.

S.7 (a)

i/p: 5,2,0,3,0

if (5>0) $\Rightarrow$ **print 5**

sum = 0+5=5

if (2>0) $\Rightarrow$ **print 2**

sum = 5+2 = 7

S.8 (c)

1st iteration:

i = 1 + 3 = 4

2nd iteration:

i = 4 + 3 = 7

Condition false. **Thus out of loop prints 7.**

S.9 (c)

Neither if condition nor else if condition is true hence **it will print the else print.**

S.11 (b)

Constant expression like 1* 2 + 4 are acceptable in cases of a switch.

S.12 (d)

int a = 1, b, dummy;

a > 10? b = 20: dummy = 1;

Note that the following would not have worked:

a > 10? b = 20 : ;;

S.13 (b, c, d)

Commant starting with /* and must end with */ combination

S.14 (b)

The input is actually a\nb. Since we are reading only two characters, **only a and \n will be read and printed.**

S.17 (c)

y *= 2 means y = y*2 i.e., y = 4, in this problem. So, the expression in equivalent to x = 4 + 4, which is 8. So, **8 will be printed.** However, the order in which the operands are evaluated is implementation-dependent. **If the right operand is evaluated first, the result will be 6.** Don't take things for granted.

S.18 (d)

9/5 yields integer 1. Printing 1 as a floating point number prints garbage.

S.19 (b)

Output

66 A

67 B

... ..

... ..

125 }

126 ~

... ..

... ..

255

256

Explanation:

The **for** loop beings with 65 and goes up till 255, printing each of these numbers along with their corresponding characters. But note that the arguments are passed to the **printf()** function from right to left. Hence the second **ch** is passed first, then it is incremented and then the first **ch** is passed. Hence when the first **ch** is passed, by that time value of **ch** already stands incremented.

Since **ch** has been declared as an **unsigned int** there is no question of exceeding the range, as the range of an **unsigned int** is 0 to 4294967295.

S.23 (c)

i % j	i = i/j	j = j + 1
true	1680	5
true	336	6
true	56	7
true	8	8
false	1	(9)

S.24 (a)

"++i" provides pre-increment, i.e. i = 6, then (++i)/(i++) provide **i = 1 with post increment resulting i = 2.**

S.26 (d)

First form the truth table of the exclusive OR operation. If both the switches are off i.e., 0, 0 then the light will be off i.e., 0. So, 0, 0 yields 0. If you switch on either of the two switches i.e., 0 1 or 1 or 1 0, the light will be on. So, 0 1 yields 1 (so does 1 0). Now, if you switch on the other one, which is currently off, it will be 1 1. This should yield a 0. Compare these results with the truth table of **XOR.**

S.27 (d)

Hexadecimal representation is 4, i.e., **m can have 4 possible values.**

S.28 (a)

(a –a/b *b%c) > a%b%c

(9 –9/5 *5%3) > 9%5%3

which is **true.**

S.29 (c)

The condition in the while loop is a must, the while loop cannot take if condition.

S.30 (b)

All statements (i), (ii) & (iii) are correct statements.

S.31 (b)

The output printed by above code is

1 1 1 1 1 1 1 1 1 1

2 2 2 2 2 2 2 2

3 3 3 3 3 3 3

4 4 4 4 4 4 ⇒ 28 (4^{th} row sum)

5 5 5 5 5

6 6 6 6

7 7 7 7 ⇒ 28 (7^{th} row sum)

8 8 8

9 9

10

Sum (4^{th} row) – Sum (7^{th} row) = 0

S.32 (d)

(a) ⇒ we can replace count as count ++ in output statement.

(b) ⇒ while is faster than do–while.

(c) ⇒ due to do–while program will run atleast once.

S.33 (b)

Sum of squares of first 10 numbers = **385**

S.35 (d)

After each pass, n is incremented by 2.

∴ sum = 1 + 9 + 25 + 49 +81

1234 + 33 + 1434 + 778 + 1203 = **4682**

S.36 (c)

1^{st} iteration **i= 1 i = 10**

2^{nd} iteration **i=2 i = 9**

3^{nd} iteration **3 8**

4^{th} iteration **4 7**

5^{th} iteration **5 6**

S.37 (b)

1.	Type declaration instructions	d.	char name, code;
2.	Input/Output instructions	c.	gets( );
3.	Arithmetic instructions	a.	k = i*234+n-7;
4.	Control instruction	b.	for(i=0;i<=10;i++);

S.38 (d)

Initially the condition n<10 in false. Hence printf will not be executed.

S.39 (d)

If choice = 'R' then the case 'R' is matched and all the case will be executed including defualt as there is no break between any cases.

S.40 (a)

The statement sum + i produces the **sum of n number starting from 1 to n.**

S.41 (d)

Return always terminates the function that executed it. main ( ) being a function, will be terminated when it executes the return statement. The return value will be returned to the calling environment, which is the operating system in this case.

S.42 (c)

k=7. So if 'k' is used as a boolean variable will be treated as a true condition. So,! k will be false i.e., so, 0<!k is actually 0<0, which is false. So, 0 will be printed.

S.46 (c)

The above declaration works properly for r, R,g,G and any other input value also.

Case 1: Suppose intput value is other than r,R,g and G then it will execute default.

Case 2 If input value is no statements corresponding to 'r' and also there is no 'break' statements.

Hence it will execute case R: printf ("RED")

Similar for case 'g' :

Case 3: If the input value is 'R' or 'G' it will execute corresponding statements.

S.50 (b)

The second assignment should be written in parentheses as follows:

a >=5? b = 100 : (b = 200);

else **always second assignment gets executed.**

S.51 (e)

variable 'j' is a character type; Hence, value of 'j' will be printed upto 127. After that for j = 128, value of character becomes negative i.e. -128 - 127 ..... 1 0 1 2 and so on, for while loop upto 255.

S.54 (b)

Output

i = 136 a = 136 k = 0

Explanation:

Observe the function call in **main()**. Since ++ precedes **i** its value is incremented to 136, and then the **!** operator negates it to give 0. This 0 is however not stored in **i** but is passed to **function()**. As a against this while evaluating the expression **!a++**, since ++ follows **a**, firstly **a** is negated to 0, this 0 is passed to **function()** and **a** is incremented to 136. Thus what get passed to **function()** are 0 and 0, which are collected in **j** and **b**, added to give another 0 and finally returned to **main()**, where it is collected in **k** and then printed out.

S.55 (b)

Output

ch doesn't contain an alphabet

Explanation:

The first and second macros have the criteria for checking whether their argument **x** is an upper or a lower case alphabet. These two criteria have been combined in the third macro, **ISALPHA**. Thus when the program goes for compilation, the **if** statement has been converted to the form:

if (ch >= 65 && ch <= 90 || ch >= 97 && ch <= 122)

As **ch** has been initialized to character +, the conditions in the **if** fail and the control rightly passes to the **else** block, from where the output is obtained.

S.56 (d)

Output

D ju zpvstfmg

Explanation:

No, your computer hasn't caught a virus! It has done just what you instructed it to. The **while** loop tests the value of the **i** element of the string. Since **i** has been initialised to 0, the first time **s[0]**, i.e. C is used. Since 'C' has a non-zero ASCII value, the condition evaluates to true, and control passes to the **if** statement. The condition to be satisfied here is that the **i** element is not a blank space. **s[0]** satisfied this condition hence the contents of the 0 elements are incremented by 1. Thus **s[0]**, i.e. 'C' having ASCII value 67. is incremented to 68, which is the ASCII value of upper vase 'D'. The new value of **s[0]** is therefore 'D'. Next **i** is incremented to 1, and the while repeats for **s[1]**. However, the **if** condition fails this time, as **s[1]** is a blank space, so this element remains unchanged. Similarly, all non-blank elements are incremented, and the **while** ends when the '\0' is reached. lastly, the **printf()** outputs the changed string.

S.57 (a)

putchar (105) will print the ASCII equivalent of 105 i.e., 'i'. The printf statement prints the current value of i, i.e., 5 and then decrements it. So, h4 will be printed in the next pass. This continues until 'i' becomes 0, at which point the loop gets terminated.

S.58 (d)

The value of count is one greate than the variable digit.

Hence when the last value of digit (digit<=9) is then the value of digit inside the loop will be 10. Hence the value of count will be one greate than count i.e. 11.

S.62 (c)

Since Madam is the required output, the function first (), should print 'a', call the function second ( ) that prints the 'd' and print 'a' again.

S.63 (c)

The preprocessor directive can be redefined anywhere in the program. So the most recently assigned value will be taken.

S.64 (d)

sum =0

for (i=1;i<=10i++)

if(i%2==0) ⇒ false

for (i = 2;2<= 10;i++)

if(2%2 ==0) ⇒True

sum = 0 + 2 = 2

for (i=3;3<=10;i++)

if (3%2==0) =false

that means, we are adding only even values to sum

∴ **sum** = 2+4+6+8+10=**30**

S.65 (a)

when j = 2

sum = 0

i =2 < 100

sum = 0+2=2 →

i = i+j = 2+2 = 4

printf ("%d",sum) →2

when j = 3

sum = 2 + 4

i = 2 + 4 = 6

when j = 4

s =2 + 4 + 6+....

Like that we find the series

2 + 4 + 6+.....+98.

S.66 (c)

initially i = 0,x=0;

0%5 == 0

∴ x = 1

then i is incremented to 1

when i = 5

5%5 = = 0

∴ x = 2

when i is incremented to 10

10 % 5 = 0

∴ x = 3

when i is incremented to 19

19 < 20

but 19% 5 ⇒0

∴ x = 4, only

when i = 20 condition in while is false

∴ control come out of loop.

x =4

∴ **Final output = 1 2 3 4 4**

S.68 (c)

initially i = 0 and i < 5

j = 0 0 ≮ 0

Now i = 1, i < 5

j = 0 and 0 < 1

{

switch (0 + 1 – 1)

case 0 : x = 0 + 1 = 1

∴ x = 1

}

Now j = 1 and 1 ≮ 1

Now, i = 2 2 < 5

j = 0, and 0 < 2

{

switch (2 + 0 – 1)

case 1:

case 2:

case 3: x = 1 + 2 = 3

∴ x = 3

default : x = 3 + 3

x = 6

Similarly final o/p as

1 6 11 16 21 24 29 32 35 38 38

S.69 (a, b, c)

The else clause has no brackets i.e., element. So, printf ("a<=b"); will be excuted anyway. if a>b or a <=b. Hence the answer.

S.70 (a)

If we enter number less than 0, then it won't accept, we have to enter number greater than zero.

In for loop; we are finding mod of that number

Let's assume, n = 121, a = n = 121

1) r = a % 10 = 121 % 10 = 1
2) x = x * 10 + r = 0 * 10 + r = 1
3) a = 121/10 = 12

   a = 12

1) r = 12 % 10 = 2
2) x = 1 * 10 + 2 = 12
3) a = 12/10 = 1

   a = 1

1) r = 1 % 10
2) x = 12 * 10 + 1 = 121
3) a = 1/10 ⇒ condition false

and finally we are comparing, (x = = n) ⇒ (121 = = 121)

i.e. **we can use the code for finding palindrome number.**

S.71 (b)

Let's assume that n = 2

In for loop, (x = 1; x <= 2; x ++) ⇒ True

In inner for loop (y = 1; y <= x; y++) ⇒ True

print y = 1 ⇒ 1

print ("\n") ⇒ control is transfer to new line

then y = y + 1 = 2 ⇒ Inner for loop condition false therefore, control is transfer to x = x + 1 ⇒ x = 2.

then also condition is true, again control goes to inner for loop and 1 2 is printed on second line again condition becomes false and finally outer for loop condition become false and control comes out. Therefore, we are getting output as

```
1
1  2
1  2  3
:  :  :
```

S.74 (b)

Due to given code, we obtain a output as follows

```
                                  1
                              1 2 3 2 1
                          1 2 3 4 5 4 3 2 1
                      1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
```

From this we can find the, **sum of the second row as = 1 + 2 + 3 + 2 + 1 = 9**

S.76 (b)

For condition checking by default, O(1) is required

if (i < 5) ⇒ True ⇒ O(2)

for (; n ≤ 4)) ⇒ True ⇒ O(2)

∴ O(n) = O(2) + O(2) + O(1) + O(1) = O(2) ⇒ By sum Rule.

S.77 (c)

Output

var = 0.000000ormat

var = 3.140000ormat

Explanation

During preprocessing, the **format** in the **printf()** statement does not get replaced by the argument **d**. Thus the **printf()** statements, after preprocessing, look like this:

printf ("var = %format\n", i);

printf ("var = %format\n", a);

This is only in keeping with waht the previous example illustrated, that the macro template within the quotes doesn't get replaced by the macro expansion. When these **printf()**s are executed, the material within quotes get dumped on to the screen as it is, except when a format specification or an escape sequence (like '\n') is encountered.

Look at the first output. Where did 'f' go? And why were the numbers printed at all? This has the simple explanation that our argument **format** happened to have as its first letter, an 'f'. The **printf()** interpreted **%f** as the format specification, and 'ormat' as something we wanted to write on the screen literally.

Note that we got the expected value for **a**, a **float**, but an absurd one for **int i**, since the **printf()** attempted to print out an **int** using **%f**.

S.78 (b)

Putchar (105) will print the ASCII equivalent of 105 i.e., 'i'. The printf statement prints the currect value of i, i.e.5 and then decrements it. some h4 will be printed in the next pass. This continues untial 'i' becomes 0, at which point the loop gets terminated.

S.79 (a)

a + b + c % 3 will be 0 if a + b + c is a multiple of 3. This will happen in one of the following ways. All three -a, b, and c are multiples of 3. This can only happen if a, b, and c take one of the 10 values, –3, 6, 9, ..., 30, independent of one another. So, there are 10 × 10 × 10 = 1000 ways this can happen. Another possibility is that a, b, and c all leave a remainder 1 so that a + b + c is evenly divisible by 3. **Considering all the different possibilities and adding, we get 9000. That will be the integer that gets printed.**

S.80 (d)

Refer S.79

The result can be analytically reasoned out. It can also be programmatically verified by having an integer variable 'counAddition' (initialized to 0) and incrementing this variable each time an addition is performed. With these changes the program fragment looks like.

```
int countAddition = 0;
d = 0;
for (i = 1; i < 31; ++i, ++ countAddition) // To account for the addition in ++i
for (j = 1; j < 31; ++j, ++ countAddition) // To account for the addition in ++j
for (k = 1, k < 31; ++k, ++ countAddition) // To acount for the addition in ++k
    if (((i + j + k) % 3) = = 0)
    {
        d = d + 1;
++ countAddition; // To account for the addition in d = d + 1
++ countAddition; // To account for the addition in i + j
++ countAddition; // To account for the addition in j + k
}
else
{
++countAddition; // To account for the addition in i + j
++countAddition; // To account for the addition in j + k
}
printf("%d", d) ;
printf ("\n%d", countAddition);
```

The value of the variable countAddition that is printed by the last statement is the answer.

S.81 (a, b)

Disadvantages of go to statement.

S.82 (b)

An arbitrary goto can lead to infinite loops as follows:

Lable 1: GOTO Label 2;

Label2: GOTO Lable 1;

Recursive porocedure/ functional calls may be non terminating if the terminating condition is never satisfied. **A repreat loop can also lead to infinite loops** if the condition to exit the loops is never satisfied.

S.83 (c)

In this, we are comparing (a>=b) & (c<b), if both are true then only we return b, that means we are finding middle number of a,b,c. Again by calling Trial function with different parameters, we are **finding middle number of a,b,c.**

S.84 (a)

If(!A[j])condition

If(A[j] = = 0)

Prints the value. But no zero is the array.

S.85 (b)

Let x = m = 9. The loop will be terminated when x – y = 0 or x – y < 0. Consider the following iteration for x = m = 9, y = 1

x–y > 0	x = (x+y)/2	y = m/x
9–1=8	x=(9+1)/2 = 5.0	y = 9/5.0=1.8
5.0–1.8=3.2	x=(5.0+1.8)/2=3.4	y=9/3.4=2.6
3.4–2.6>0	x=(3.4+2.6)/2=3	y=9/3.0=3.0

3.0 – 3.0 = 0 loop terminated

So, m = 9 then x = 3

$(m)^{1/2} = (3)^{1/2}$

$m = 9$

$\Rightarrow x = 3$

So the algorithm compute $m^{1/2}$.

S.86 (d)

Loop invariant is the part of code motion. Loop invarient for while loop is a condition, if we assign this condition before the while loop then there is no effect in the code and produces same output

Consider the given code

```
int n, rev;
  rev. = 0;
while ( n > 0) {
rev = rev * 10 + n%10;
  n = n/10;
}
```

Let input n = 12345

where $d_1 = 1$, $d_2 = 2$, $d_3 = 3$, $d_4 = 4$ and $d_5 = 5$

Iteration	n = n/10	rev = rev * 10 + n%10
0	$d_1 d_2 d_3 d_4 d_5$ 1 2 3 4 5	rev =0
1	$d_1 d_2 d_3 d_4$ 1 2 3 4	d_5 rev =0*10+5=5
2	$d_1 d_2 d_3$ 1 2 3	$d_5 d_4$ rev =5*10+4=5 4
3	$d_1 d_2$ 1 2	$d_5 d_4 d_3$ rev =54*10+3=5 4 3
4	d_1 1	$d_5 d_4 d_3 d_2$ rev =543*10+2=5 4 3 2
5	0	$d_5 d_4 d_3 d_2 d_1$ rev =5432*10+5=5 4 3 2 1

Let $n = d_1 d_2 \ldots d_m$ in the i^{th} iteration rev = $d_m d_{m-i} + 1$

For example if m = 5 and i = 3 the above example produces

$n = d_1 d_{5-3}(d_2)$ $\quad$ $rev = d_5 d_4 d_{5-3+1}(d_3)$

1 2 $\quad$ 5 4 3

S.87 (c)

```
1. int f(int n)
2. {  static int i = 1 ;
```