MADE EASY PUBLICATIONS

POSTAL BOOK PACKAGE 2027

COMPUTER SCIENCE & IT

OBJECTIVE PRACTICE SETS VOLUME-IV

CONTENTS

ALGORITHMS

OBJECTIVE PRACTICE SETS

Page No. 1 - 102

1 CHAPTER
Asymptotic Analysis of Algorithms

Multiple Choice Questions

Q.1 The concept of order (Big O) is important because
(a) It can be used to decide the best algorithm that solves a given problem
(b) It determines the maximum size of a problem that can be solved in a given amount of time
(c) It is the lower bound of the growth rate of algorithm
(d) Both (a) and (b) above

Q.2 Let $f(n) = \Omega(n)$ and $g(n) = \Omega(n^2)$. Then $f(n) + g(n)$ is
(a) $\Omega(n)$ (b) $\theta(n)$
(c) $\Omega(n^2)$ (d) $O(n)$

Q.3 The order of an algorithm that finds whether a given Boolean function of 'n' variables, produces a 1 is
(a) Constant (b) Linear
(c) Logarithmic (d) Exponential

Q.4 $f(n) = 5n^2 + 6n + 10$
Which will be the exact value for $f(n)$?
(a) $\theta(n^2)$ (b) $O(n^2)$
(c) $o(n^2)$ (d) $\Omega(n^2)$

Q.5 Match the following groups:

	Group-I ($n > 0$)	Group-II
A.	$3n + 4n^2 + 5n \log n$	1. $O(1)$
B.	$n + \log n + \log \log n$	2. $O(\log n)$
C.	$10 + n + n \log n + \log n$	3. $O(n)$
D.	$10 + 10000 + 100000$	4. $O(n \log n)$
		5. $O(n^2)$

Codes:

	A	B	C	D
(a)	5	3	4	2
(b)	5	4	3	1
(c)	5	3	4	1
(d)	5	4	3	2

Q.6 Let $f(n) = \Omega(n)$ and $g(n) = \Omega(n^2)$. Then $f(n) + g(n)$ is
(a) $\Omega(n)$ (b) $\theta(n)$
(c) $\Omega(n^2)$ (d) $O(n)$

Q.7 Find which of the following is not correct?
(a) $\sum_{i=1}^{n} \sqrt{i} = O(n^{3/2})$
(b) $n^2 \log n = \Theta(n^2)$
(c) $100n^3 + 2n^2 = \Omega(n^2)$
(d) $n! = O(n^n)$

Q.8 Unrestricted use of Goto is harmful, because it
(a) increase running time of programs.
(b) makes debugging difficult.
(c) results in the compiler generating longer machine code.
(d) increase memory requirement of programs.

Q.9 Each of the function $2^{\sqrt{n}}$ and $n^{\log n}$ has a growth rate .... than that of any polynomial.
(a) Greater (b) Less
(c) Equal to (d) Uncertain

Q.10 Consider the following function:

```
void f(int n)
{
    int i, j, k, m;
    for (i = 0; i < 100; i ++)
    {
        for (j = 0; j < n; j++)
        {
            for (k = 0; k < j; k++)
                printf("%d", k);
        }
    }
}
```

What is the worst case running time of the function f for any positive value of n?
(a) $O(1)$ (b) $O(n)$
(c) $O(n^2)$ (d) $O(n^3)$

Q.11 Consider the following function:

```
int unknown (int n)
{
    int i, j, k = 0;
    for (i = n/2; i < = n; i++)
        for (j = 2; j <= n; j = j* 2)
            k = k + n/2;
    return (k);
}
```

The return value of the function is

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$ (d) $\Theta(n^3 \log n)$

Q.12 Consider the following segment of c-code:

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

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

(a) $\lceil \log_2 n \rceil + 1$ (b) n

(c) $\lceil \log_2 n \rceil$ (d) $\lfloor \log_2 n \rfloor + 1$

Q.13 In the following C function, let $n \geq m$.

```
int gcd (n, m)
{   if (n% m = = 0} return m;
    n = n% m;
    return gcd (m, n);
}
```

How many recursive calls are made by this function?

(a) $\Theta(\log_2 n)$ (b) $\Omega(n)$

(c) $\Theta(\log_2 \log_2 n)$ (d) $\Theta\left(\sqrt{n}\right)$

Q.14 What is time complexity of following code:

```
int a = 0;
for (i = 0; i < N; i++)
{
    for (j = N; j > i; j--)
    {
        a = a + i + j;
    }
}
```

(a) $O(N)$ (b) $O(N * (\log N)$

(c) $O(N * \text{Sqrt}(N))$ (d) $O(n^2)$

Q.15 What does it mean when we say that one algorithm X is asymptotically more efficient than Y?

(a) X will always be a better choice for small inputs.

(b) X will always be a better choice for large inputs.

(c) Y will always be a better choice for small inputs.

(d) X will always be a better choice for all inputs.

Q.16 What is time complexity of following code:

```
int a = 0, i = N;
while (i > 0) {
    a = a + i;
    i/ = 2;
}
```

(a) $O(N)$ (b) $O(\text{Sqrt}(N))$

(c) $O(N/2)$ (d) $O(\log N)$

Q.17 What is time complexity of fun( )?

```
int fun (int n)
{
    int count = 0;
    for (int i = n; i > 0; i/ = 2)
      for (int j = 0; j < i; j++)
      count + = 1;
    return count;
:
```

(a) $O(n^2)$ (b) $O(n)$

(c) $O(n \log n)$ (d) $O(n (\log n)^2)$

Q.18 Consider the following recursive function:

```
int F (int array [ ], int n)
{
    int S = 0;
    if (n = = 0)
        return 0;
    S = F (array, n – 1)
    if (array [n – 1] < 0)
        S = S + 100;
    return S;
}
```

What is the worst case time complexity of the above function?

(a) $O(n)$ (b) $O(n \log n)$

(c) $O(n^2)$ (d) $O(\log n)$

Q.19 What is time complexity of following program?

```
Void fun (int n){
int i, j, counter = 0;
for (i = 1; i ≤ n; i++) {
for (j = 1; j × j ≤ n; j++) count++;}}
```

(a) $O(n^{2)}$ (b) $O(n^{3/2})$
(c) $O(n \log n)$ (d) $O(n \log \log n)$

Q.20 Consider the following functions:

1. $n!$
2. a^n, a is constant, $a > 0$
3. n^n
4. n^k, k is a constant, $k > 0$
5. e^n

Choose the correct statement which ranks all functions by order of growth.

(a) $2 < 4 < 5 < 1 < 3$
(b) $4 < 2 < 3 < 5 < 1$
(c) $4 < 5 < 2 < 1 < 3$
(d) $4 < 2 < 5 < 1 < 3$

Q.21 Consider the following functions:

n, $\log n$, $\sqrt{n}$, $\log(\log n)$, $\frac{n}{\log n}$, $(\log n)^2$,

$\sqrt{n}\log n$, $n\log n$

Identify the functions in increasing order of growth.

(a) $\frac{n}{\log n}$, $\log(\log n)$, $(\log n)$, $(\log n)^2$,

$\sqrt{n}$, $\sqrt{n} \cdot \log n$, n

(b) $\log(\log n)$, $\log n$, $(\log n)^2$, $\sqrt{n}$,

$\sqrt{n}\log n$, $\frac{n}{\log n}$, n, $n \log n$

(c) $\log(\log n)$, $\log n$, $(\log n)^2$,

$\sqrt{n}$, $\frac{n}{\log n}$, $\sqrt{n}\log n$, n, $n \log n$.

(d) None of these

Q.22 Which one of the following is true?

1. $an = O(n^2)$ (small oh) $a \geq 0$
2. $an^2 = O(n^2)$ (big oh) $a > 0$
3. $an^2 \neq O(n^2)$ (small oh) $a > 0$

(a) Only 1 and 2 are correct
(b) Only 1 is correct
(c) 1 and 3 are correct only
(d) All are correct

Q.23 Consider the following statements:

1. Any two functions f, g are always comparable under big-oh that is $f = O(g)$ or $g = O(f)$
2. If $f = O(g)$ and $f = O(h)$ then, $g(n) = \theta(h)$

Select correct option:

(a) 1 is true and 2 is false
(b) 1 is false and 2 is true
(c) Both are false
(d) Both are true

Q.24 1. $\frac{1}{2}n^2 = \omega(n)$, 2. $\frac{1}{2}n^2 = \omega(n^2)$

Which of the following is true?

(a) 1 is correct
(b) 2 is correct
(c) 1 and 2 both are correct
(d) None of these

Q.25 Consider the following functions:

$f(n) = 2^{\log_2 n}$
$f(n) = n^{\log_2 n}$
$h(n) = n^{1/\log_2 n}$

Which of the following statements about the asymptotic behaviour of $f(n)$, $g(n)$ and $h(n)$ is true?

(a) $f(n) = \Omega(g(n))$ and $g(n) = O(h(n))$
(b) $g(n) = \Omega(h(n))$ and $f(n) = O(f(n))$
(c) $f(n) = O(g(n))$ and $g(n) = \Omega(h(n))$
(d) $g(n) = O(h(n))$ and $h(n) = O(g(n))$

Q.26 Suppose $f, g, h, k : N \to N$.
If $f = O(h)$ and $g = O(k)$, then

(a) $f + g = O(h + k)$
(b) $fg = (hk)$
(c) Both (a) and (b) above
(d) None of the above

Q.27 Let $g(n) = \Omega(n)$, $f(n) = O(n)$ and $h(n) = \theta(n)$ then what is the time complexity of $[g(n)\ f(n) + h(n)]$

(a) $O(n)$ (b) $\theta(n)$
(c) $\Omega(n)$ (d) $\theta(n^2)$

Q.28 Consider an array of n element with sorted order, if any element i appear more than half the number of element, what is the time complexity to count the number of occurrences of i?

(a) $O(\log n)$ (b) $O(1)$
(c) $O(n)$ (d) $O(\log\log n)$

Q.29 Find the time complexity of the following summation. Assume that k is a constant, $k > 0$

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{k}$$

(a) $O(n)$ (b) $O(n^2)$
(c) $O(n^3)$ (d) None of these

Answers Asymptotic Analysis of Algorithms

1.	(d)	2.	(c)	3.	(d)	4.	(b)	5.	(c)	6.	(c)	7.	(b)	8.	(b)	9.	(a)
10.	(c)	11.	(b)	12.	(d)	13.	(a)	14.	(d)	15.	(b)	16.	(d)	17.	(b)	18.	(a)
19.	(b)	20.	(d)	21.	(b)	22.	(c)	23.	(c)	24.	(a)	25.	(c)	26.	(c)	27.	(c)
28.	(a)	29.	(b)	30.	(a)	31.	(d)	32.	(d)	33.	(d)	34.	(c)	35.	(a)	36.	(c)
37.	(a)	38.	(a)	39.	(a)	40.	(c)	41.	(b)	42.	(c)	43.	(b)	44.	(c)	45.	(a)
46.	(a)	47.	(d)	48.	(d)	49.	(d)	50.	(a)	51.	(a)	52.	(a)	53.	(c)	54.	(d)
55.	(b)	56.	(d)	57.	(a)	58.	(c)	59.	(d)	60.	(c)	61.	(b)	62.	(b)	63.	(a)
64.	(b)	65.	(d)	66.	(b)	67.	(b)	68.	(a)	69.	(c)	70.	(c)	71.	(c)	72.	(c)
73.	(c, d)			74.	(a, b)			75.	(a, b, d)			76.	(a, c)				

Explanations Asymptotic Analysis of Algorithms

1. (d)

Big O notation gives worst case limit for a given problem. Also find out the least upper bound of problem.

2. (c)

Given

$$f(n) = \Omega(n)$$

i.e. $f(n) \geq c_1(n)$

$f(n)$ can be anything but atleast (n) not less than (n)

Given

$$g(n) = \Omega(n^2)$$

i.e. $g(n) \geq c_2(n^2)$

$g(n)$ can be anything but atleast (n^2) not less than (n^2)

$$f(n) + g(n) = \Omega((n) + (n^2)) = \Omega(n^2)$$

Here we can not comment about upper bound.

3. (d)

In the worst case it has to check all the 2^n possible input combinations, which is exponential.

4. (b)

$$f(n) = 5n^2 + 6n + 10$$

So, $\underset{\text{(Big–oh)}}{O(n^2)}$ is exact value of $f(n)$.

So, answer is (b).

5. (c)

$3n + 4n^2 + 5n \log n = O(n^2)$

$n + \log n + \log \log n = O(n)$

$10 + n + n \log n + \log n = O(n \log n)$

$10 + 10000 + 100000 = O(1)$

6. (c)

Given

$$f(n) = \Omega(n)$$

i.e. $f(n) \geq c_1(n)$

$f(n)$ can be anything but atleast (n) not less than (n)

Given

$$g(n) = \Omega(n^2)$$

i.e. $g(n) \geq c_2(n^2)$

$g(n)$ can be anything but atleast (n^2) not less than (n^2)

$$f(n) + g(n) = \Omega((n) + (n^2)) = \Omega(n^2)$$

Here we can not comment about upper bound.

7. (b)

$n^2 \log n \leq k.n^2$ will not satisfy for any constant k.

$\therefore$ Option (c) is not correct.

8. (b)

Unrestricted we of goto statement is harmful because it makes more difficult to verifying programs i.e., use of goto can results in unstructured code and there can be blocks with multiple entry and exit which can cause difficulty which debugging of program.

9. (a)

$2^{\sqrt{n}}$ and $n^{\log n}$ grows exponentially which have growth rate greater than any polynomial.

10. (c)

$$f(n) = \sum_{i=0}^{99} \sum_{j=0}^{n-1} \left(\sum_{k=0}^{j-1} 1 \right) = O(n^2)$$

11. (b)

Outer loop execute for $\frac{n}{2}+1$ iterations. Inner loop executes for $\log_2 n$ iterations. In every iteration of inner loop $\frac{n}{2}$ is added to k.

Return value = $\frac{n}{2}$ × number of outer loops × number of inner loops

$$= \frac{n}{2} \times \left(\frac{n}{2} + 1 \right) (\log n)$$

$$= O(n^2 \log n)$$

12. (d)

Let the increment of j is 2^0, 2^1, ... 2^i for some value of i so, according to the question for while loop; $2i \le n$ or $i \le \log_2 n$.

One extra comparison required for the termination of while loop.

So, total number of comparisons

$$= i + 1 = \lfloor \log_2 n \rfloor + 1.$$

13. (a)

Let, $T(m, n)$ be the total number of steps.

So, $T(m, 0) = 0$, $T(m, n) = T(n, m \bmod n)$ on average

$$T_n = \frac{1}{n} \sum_{0 \le k \le n} T(k, n)$$

$$T_n \approx 1 + \frac{1}{n} (T_0\ T_1 + \ldots + T_{n-1})$$

$$T_n \approx S_n$$

$$S_n = 1 + \frac{1}{n} (S_0\ S_1 + \ldots + S_{n-1})$$

$$S_n = 1 + \frac{1}{n+1} (S_0\ S_1 + \ldots + S_n)$$

$$= 1 + \frac{1}{n+1} (n(S_{n-1}) + S_n)$$

$$= 1 + \frac{1}{n+1}$$

$$= S_n + \frac{1}{n+1}$$

So, $T_n \approx \Theta(\log_2 n) + 0(1)$

$$T \approx \Theta(\log_2 n)$$

14. (d)

The above code runs total number of times

$= N + (N-1) + (N-2) + \ldots + 1 + 0$

$= N * (N+1)/2$

$= \frac{1}{2} * N \wedge 2 + \frac{1}{2} * N = O(N \wedge 2)$ times

15. (b)

In asymptotic analysis, we consider growth of algorithm in terms of input size. An algorithm X is said to be asymptotically better than Y if X takes smaller time than Y for all input sizes n larger than a value n_0 where $n_0 > 0$.

16. (d)

We have to find smallest x such that $N/2 \wedge X\ N$

$$X = \log(N)$$

So, $O(\log N)$ is time complexity.

17. (b)

For n time, inner loop will execute for n times.

For $\frac{n}{2}$ time, inner loop will execute for $\frac{n}{2}$ times.

For $\frac{n}{4}$ time, inner loop will execute for $\frac{n}{4}$ times

and do on ...

So, time complexity:

$$T(n) = O\left(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1 \right) = O(n)$$

18. (a)

Recurrence relation of function F

$F(n) = 0$ if $n = 0$

$F(n) = F(n-1) + 1$, $n > 0$

Time complexity = $O(n)$

19. (b)

for ($i = 1$; $i \le n$; i++) $\Rightarrow$ $O(n)$

for ($j = 1$; $j \times j \le n$; j++) $\Rightarrow$ $O(\sqrt{n})$

count ++;

Total time complexity = $O(n \times n^{1/2}) = O(n^{3/2})$

20. (d)

$n^k < a^n < e^n < n! < n^n$

n^n will take maximum asymptotic time.

$n! = 0(n^n)$

$e < n \quad \therefore e^n < n^n$

$k < n \quad \therefore n^k < a^n$

21. (b)

$\log(\log n) < \log n < (\log n)^2 < \sqrt{n} <$

$\sqrt{n}\log n < \dfrac{n}{\log n} < n < n \log n$

So option (b) is correct.

22. (c)

1. $an = O(n^2)$ for $a \geq 0$
   True
2. $an^2 = O(n^2)$ for $a > 0$
   False, big-*oh* needed.
3. $an^2 \neq O(n^2)$ for $a > 0$
   True, big-*oh* needed

So, 1 and 3 are correct.

23. (c)

Both are false.

Statement 1 is false.

Reason: Consider $f(n) = 0.5$ and $g(n) = \sin(n)$ or, consider $f(n) = n$ and $g(n) = 1$ when n is even; n^2 when n is odd.

In both the above cases neither $f(n) = O\, g(n)$ nor $g(n) = Of(n)$

Statement (2) is false.

Reason: Consider $g(n) = 2n$ and $h(n) = n$ and $f(n) = 10n$

24. (a)

1. $\dfrac{1}{2}(n^2) = \omega(n)$

   $\dfrac{1}{2}(n^2) \geq C_1(n)$

   So true always

2. $\dfrac{1}{2}(n^2) = \omega(n^2)$

   $\dfrac{1}{2}(n^2) \geq C_1(n^2)$

   So false, since $\Omega(n^2)$ is true.

25. (c)

$f(n) = 2^{\log_2 n} = n^{\log_2 2} = n$

$g(n) = n^{\log_n}$

$h(n) = n^{\frac{1}{\log n}} = \sqrt[\log n]{n}$

$[n > \sqrt[\log n]{n}$ for all large value of $n]$

[it is less than n since max power of n is always less than 1 for large value of n.]

So, $g(n) \geq f(n) \geq h(n)$

So, $f(n) = O(g(n))$ and $g(n) = \Omega(h(n))$

So, option (c) is correct.

26. (c)

If $f = O(h)$

and $g = O(k)$

Then, $(f + g) = O(n + k)$

and $(f \times g) = O(nk)$

27. (c)

$g(n) = \Omega(n) \quad g(n) \geq C_1 \cdot n$

$f(n) = O(n) \quad f(n) \leq C_2 n$

$h(n) = \theta(n) \quad C_3 \cdot n \leq h(n) < C_4 \cdot n$

$g(n) \cdot f(n) + h(n)$

$\geq C \cdot n \quad \theta(n)$

$= \Omega(n)$

28. (a)

Time complexity is $O(\log n)$ to count number of occurrences of i.

29. (b)

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(\frac{1}{k}\right) = \frac{1}{k}\sum_{i=1}^{n}\sum_{j=i+1}^{n}(1)$$

$$= \frac{1}{k}\sum_{i=1}^{n}\left[1+1+1+\ldots+n-(i+1)+1 \text{ times}\right]$$

$$= \frac{1}{k}\sum_{i=1}^{n}[n-i]$$

$$= \frac{1}{k}\left[n\sum_{i=1}^{n}(1) - \sum_{i=1}^{n}(i)\right] = \frac{1}{k}\left[n\cdot n - \frac{n(n+1)}{2}\right]$$

$$= \frac{1}{k}\left[n^2 - \frac{n^2+n}{2}\right] = \frac{1}{2k}\left[n^2 - n\right] = O(n^2)$$

30. (a)

```
ing count = 0, N; ....O(1)
for (i = 0; i < N * 2; i++) .....O(N)
{
    for(j = 0; j < i/2; i++) .....O(N/3)
    {
        for(k = 0; k < j * j; k++).....O((n * n)/3)
        {
            count ++;
```

PROGRAMMING & DATA STRUCTURES

OBJECTIVE PRACTICE SETS

Page No. 103 - 205

1 CHAPTER
Programming Methodology

Multiple Choice Questions & NAT Questions

1. Consider the following function declaration

int* *f*(int *);

Which of the following is correct about the declaration?

(a) *f* is a function which takes integer pointer as argument and returns integer.
(b) *f* is a function which takes integer pointer as an argument and returns address of an integer.
(c) *f* is a pointer to a function which takes integer pointer as an argument and returns integer.
(d) *f* is a pointer to a function which takes integer pointer as an argument and returns address of an integer.

2. Find the output of the following program:

```
main( )
{
    extern int i;
    i = 20;
pirntf("%d", i);
}
```

(a) Linker error (b) 20
(c) Compiler error (d) None of these

3. Consider the following code?

```
void main( )
{   static int i = 5;
    if (– – i)
    {
        main ( );
        printf ("%d", i);
    }
}
```

How many zero's are printed in the output?

4. Which of the following is correct output for the program code given below?

```
main( )
{
    void pr( );
    pr ( );
    pr ( );
    pr ( );
}
void pr ( )
{
    static int i = 1;
    printf ("%c", (65+ i ++));
}
```

(a) 66, 67, 68 (b) 66, 66, 66
(c) 67, 68, 69 (d) None of these

5. Which of the following are equivalent to the statement?

int *k* = (*i* << 3) + (*j* >> 2)

(a) int *k* = *i* * 8 + *j*/4;
(b) int *k* = *i* * 3 + *j* * 2;
(c) int *k* = *i* * 3 + *j*/2;
(d) int *k* = *i*/8 + *j* * 4;

6. Consider the following foo function and identify the return value of foo function.

```
int foo (unsigned int n)
{   int c, x = 0;
    while (n! = 0)
    {   if (n & 1) x ++;
        n >>= 1;
    }
    return c;
}
```

(a) It counts the total number of bits set in an unsigned integer.
(b) It counts the number of bits which are zero.
(c) It counts the number of occurrences of 01.
(d) It returns the same value as '*n*'.

7. Consider the following code:

```
int f(int a, int b)
{
    if (b == 0) return 1;
    else if (b % 2 == 0)
    {
        return (f(a, b/2) * f(a, b/2));
    }
    else
    {
        return (a * f(a, b/2) * f(a, b/2));
    }
}
```

The return value of $f(2, 10)$ is _____.

8. What is output of the following program?

```
# include <stdio.h>
# define R 10
# define C 20
int main( )
{    int (*P) [R] [C];
     printf("%d", size of (*P));
     getchar( );
     return 0;
}
```

(a) 4 (b) 8
(c) 2 (d) None of these

9. Match **List-I** with **List-II**:

List-I

A. typedef int (* ptr) ( ); ptr *p*;
B. int (* *P*) [4];
C. int * *P* [4];

List-II

1. Pointer to an array of integer
2. Pointer to a function returning an integer
3. Array of pointers, pointing to integer

Codes:

	A	B	C
(a)	1	2	3
(b)	2	1	3
(c)	2	3	1
(d)	1	3	2

10. Consider the following pseudocode program:

```
int i
main ( )
{
    i = 3
    S ( )
    R ( )
}
void S ( )
{
    print i // prints the value of i on the current line of output
    print " " // prints a blank space on the current line of output
}
void R ( )
{
    int i
    i = 2
    S ( )
}
```

What is the output of the program if the pseudocode uses either static (lexical) scoping or dynamic scoping?

	Static Scoping	Dynamic Scoping
(a)	3 2	3 2
(b)	3 3	2 2
(c)	3 3	2 3
(d)	3 3	3 2

11. Consider the following code:

```
int a = 32, b = 2, c = 3;
Switch (X)
{
    Case 2: printf("%d", a);
    Case 4: printf("%d", b);
    Case 6: break;
    Case 8: printf("%d", c);
    default: printf("%d", b);
}
```

Find the missing statement *X*, if the above '*C*' code prints the output as 32.

(a) $b * c$ (b) $b * c - 2$
(c) $b + c * 2$ (d) None of these

12. Which of the following statement is false about 'return' statement?

(a) It terminates the execution of a function.

(b) Control moves back to the calling environment after the return statement execution.

(c) It cannot contain an expression.

(d) It may appear more than once in the same function.

13. Consider the following pseudocode:

```
int i = 0;
main( )
{
    i = 3;
    A( );
    B( );
}
A( ) { print "i"; }
B( ) { int i = 2; A( )}
```

What is the output of the above code if it uses static scoping?

(a) 2, 3 (b) 3, 2

(c) 2, 2 (d) 3, 3

14. Which of the following is a valid switch statement?

(a)
```
switch (i)          // i is an integer
{
    case 1: break;
    case j: break; // j is a variable
}
```

(b)
```
switch (i)          // i is a string
{
    case "abc" : break;
    case "xyz" : break;
}
```

(c)
```
switch (i)          // i is an integer
{
    case 1 : break;
    case 2*4 : break;
}
```

(d) Both (a) and (c)

15. Consider the following code:

```
int main( )
{
    char A[ ] = "gate";
    int x;
    for (x = 0; A[x]; x++)
    {
        printf("%c", A[x]);
    }
}
```

What is the output printed by the code?

(a) gate (b) g

(c) run time error (d) compile time error

16. What will be the output of the following program?

```
#include <stdio.h>
#include <string.h>
int main( )
{
    int X = size of ("MADEEASY");
    int Y = strlen ("MADEEASY");
    printf("%d%d", X, Y)
    return 0;
}
```

(a) 88 (b) 99

(c) 89 (d) 98

17. Consider the following C program:

```
# include <stdio.h>
void f(int x, int * p)
{
    *p = x;
    x = 10;
}
int main( )
{
    int a = 5,  b = 6;
    int  *p = &a, **q;
    *p = 20; q = &p;
    f(a, &b);
    *q = &b;
    *p = 30;
    printf("%d, %d", a, b);
}
```

What is the output product by above C program?

(a) 10, 20 (b) 20, 30

(c) 30, 10 (d) 20, 20

18. What will be the output printed by the following C program

```
void main( )
{
    int x = 1,  i,  y = 2;
    for (i = 0; i < 5;  i++)
    {
        x << 1;
        y = x + i;
    }
    printf("%d, %d", x, y);
}
```

(a) 1, 5 (b) 32, 5
(c) 1, 72 (d) 32, 72

19. Which of the following is illegal statement in C.
(a) int (**p) [ ]; (b) int*(*p) ( );
(c) int (*f( )) [ ]; (d) int*f( ) [ ];

20. Consider the following recursive C functions:

```
int f(int i)
{
    if(x == 0) return 1;
    return f(x – 1) + g(x – 1);
}
int g(int x)
{
    if (x == 0) return 2;
    return g(x – 1) + g(x – 1);
}
```

What is the value returned by f(g(1))?

21. Which of following declarations represents an array of *N* pointers to functions, returning pointers to functions and returning pointer to character?
(a) char **((*a[N]( )) ( );
(b) char **((*a[N]))( )( );
(c) char ***((a[N]( ))) ( );
(d) char *(*(*a[N])( )) ( );

22. What is the output of the following code:

```
void main( )
{
    int const*p = 5;
    printf("%d", ++(*p));
}
```

(a) 5 (b) 6
(c) 7 (d) Compiler error

23. Consider the following rec function:

```
rec (int x)
{
    static int f;
    if (x ==1)
        return (1);
    else
        f+ = x * rec (x – 1);
    return (f);
}
```

Find the value returned by rec (5).

24. Find the output of the following program:

```
main( )
{   int i =_1_abc (10);
    printf("%d \ n", – –i);
}
int  _1_abc (int i)
{
    return (i++);
}
```

25. What is the value returned by the following function when x =1 and y = 3?

```
int fun (int x, int y)
{
   if (x == 0 && y > = 0) return y + 1;
   else if (x > 0 && y == 0) return f(x – 1, 1);
   else if (x > 0 && y > 0) return (f(x – 1, f(x, y – 1)));
}
```

26. What does the following fragment of C program print?

```
char x[ ] = "JSHAKZAAOHE";
char *y = x;
printf("%s", x + y [10] – y[7]);
```

(a) Prints the entire string
(b) Prints only "AKZAAOHE"
(c) Prints only "KZAAOHE"
(d) Prints only "AAOHE"

27. Consider the following code:

```
int Do (char *gate)
{
    char *gate1 = gate;
    char *gate2 = gate + strlen (gate) – 1;
    while (gate1 < gate)
```

Answers Programming Methodology

1.	(b)	2.	(a)	3.	(4)	4.	(d)	5.	(a)	6.	(a)	7.	(1024)	8.	(d)	9.	(b)		
10.	(d)	11.	(c)	12.	(c)	13.	(d)	14.	(c)	15.	(9)	16.	(d)	17.	(b)	18.	(a)		
19.	(d)	20.	(21)	21.	(d)	22.	(d)	23.	(240)	24.	(9)	25.	(5)	26.	(c)	27.	(c)		
28.	(d)	29.	(a)	30.	(a)	31.	(b)	32.	(a)	33.	(114)	34.	(c)	35.	(50)	36.	(10)		
37.	(a)	38.	(b)	39.	(50)	40.	(115)	41.	(43211234)	42.	(c)	43.	(d)	44.	(b)				
45.	(a)	46.	(c)	47.	(a)	48.	(b)	49.	(b)	50.	(c)	51.	(c)	52.	(c)	53.	(b)		
54.	(d)	55.	(c)	56.	(d)	57.	(65)	58.	(13)	59.	(a)	60.	(40)	61.	(b)	62.	(a)		
63.	(a)	64.	(c)	65.	(166)	66.	(c)	67.	(c)	68.	(290)	69.	(1365)	70.	(10)	71.	(d)		
72.	(51)	73.	(23)	74.	(c)	75.	(61)	76.	(d)	77.	(b)	78.	(15)	79.	(0)	80.	(300)		
81.	(c)	82.	(50)	83.	(c)	84.	(556)	85.	(a)	86.	(b)	87.	(a)	88.	(302011)				
89.	(d)	90.	(106)	91.	(d)	92.	(17)	93.	(b)	94.	(b)	95.	(d)	96.	(b)	97.	(d)		
98.	(a, b, c)	99.	(a, c)	100.	(b)														

Explanations Programming Methodology

1. (b)

The correct declaration for (a) is int *f*(int ∗)

The correct declaration for (b) is int* *f*(int *);

The correct declaration for (c) is int (∗*f*) (int ∗)

The correct declaration for (d) is int ∗(∗*f*) (int ∗)

2. (a)

Linker error: Undefined symbol-*i*

Extern int *i*; Specifies to the compiler that the memory for *i* is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name '*i*' is available in any other program with memory space allocated for it. Hence linker error occurred.

3. (4)

The variable '*i*' is declared as static, hence memory for '*i*' will be allocated for only once, as it encounters the statement. The function main( ) will be called recursively unless *i* becomes equal to zero and since main ( ) is recursively called, so the value of static *i*, i.e. 0 will be printed every time the control is returned.

So total 4 times zero is printed.

4. (d)

The correct output is "*BCD*" when the function pr ( ) is first called the value of *i* is initialized to 1. After the pr ( ) completes its execution *i* = 2 is retained for it's next call as "*i*" is static variable.

∴ 65 + 1 = 66 (B)

65 + 2 = 67(C)

65 + 3 = 68(D)

∴ *BCD* is the correct output.

5. (a)

<< and >> are bit wise operators used to multiply and divide by power of 2 respectively (shift operators)

∴ $i << 3 \Rightarrow i * 8$

$j >> 2 \Rightarrow j / 4$

6. (a)

It counts the number of bits set in an unsigned integer.

```
while (n! = 0)
{
    if (n & 1) x ++;
    /* performs bit wise AND operator and if
    condition is satisfied if result
    contains atleast one 1.
```

$n >> = 1$

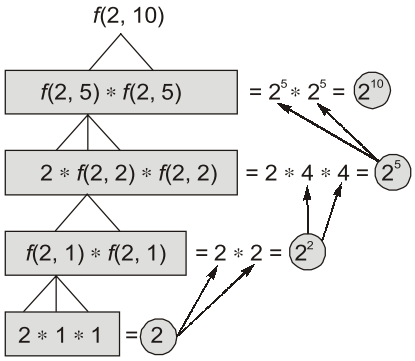
}

x ++; Maintains the count for number of 1's.

n >>= 1 Shift the 'n' bit number by 1 bit to right.

7. (1024)

$f(2, 10)$ returns 2^{10} value = 1024



8. (d)

int (*p) [R] [C] ⇒ pointer to an array of array of integer.

Output: 10 * 20 * size of (int) which is 800 for compliers with integer size as 4 bytes and 400 for compilers with integer size as 2 bytes.

The pointer p is de-referenced, hence it yields type of the object. In the present case, it is an array of array of integers.

So, it prints R * C * size of (int).

9. (b)

A : return type is int. It is a pointer to a function.

B : (* P) declares pointer. (* P) [4] is array pointed by pointer.

C : * P [4] declares array of pointers.

10. (d)

Using static scoping: First print prints the global i whose value is 3. Second print prints the global i whose value is 3.

Using dynamic scoping: First print prints the global i whose value is 3. Second print prints the local i whose value is 2 (from the function it was called).

11. (c)

X : $b + c * 2$ is 8

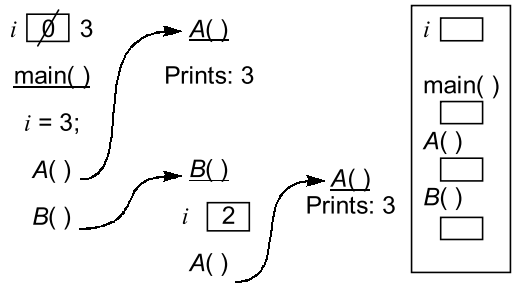
Case 8: prints 3 then default case prints 2

∴ Output prints 32.

12. (c)

Return statement can contain an expression.

13. (d)



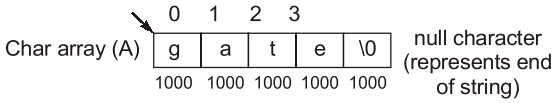
Output printed by the code: 3, 3

14. (c)

Only constants or enums can be used with cases of switch. 2 * 4 is a constant expression.

15. (a)

Let string gate be stored from memory location 1000.



Given loop prints string from A[0] to A[3], i.e., "gate"

16. (d)

Size of ( ) returns length of string including null character (\°). While strlen ( ) returns length of string without including null character.

So here output is $X = 9$, $Y = 8$.

17. (b)

After Execution

main ( )	a	b	p	q
int a = 5, b = 6;	5	6		
int *p = &a, **q;	5	6	&a	–
*p = 20; q = &p;	20	6	&a	&p
f(a, &b);	20	20	&a	&p
*q = &b;	20	20	&b	&p
*p = 30;	20	30	&b	&p

$a = 20$, $b = 30$

18. (a)

Iterations of for statement

for :	$i=0$		$i=1$		$i=2$		$i=3$		$i=4$	
	x	y	x	y	x	y	x	y	x	y
$x << 1;$	1	2	1	1	1	2	1	3	1	4
$y = x + i;$	1	1	1	2	1	3	1	4	1	5

$x = 1,\ y = 5$

Note: $x << 1$ will not change the value of x, but $x = x << 1$ will changes the value of x.

19. (d)

(a) int (**p) [ ];
A pointer to a pointer to an array of integers.

(b) int *(*p) ( );
A pointer to a function returning an integer pointer.

(c) int (*f ( )) [ ];
A function returning a pointer to an array of integers.

(d) int *f( ) [ ]; is illegal statement
A function returning an array of int pointers.

20. (21)

$f(g(1))$:

$$g(1) = g(0) + g(0) = 2 + 2 = 4$$
$$\Rightarrow f(g(1)) = f(4) = f(3) + g(3)$$
$$f(3) = f(2) + g(2)$$
$$f(2) = f(1) + g(1)$$
$$f(1) = f(0) + g(0)$$
$$f(0) = 1$$
$$g(0) = 2$$
$$g(1) = g(0) + g(0) = 4$$
$$g(2) = g(1) + g(1) = 8$$
$$g(3) = g(2) + g(2) = 16$$
$$\Rightarrow f(0) = 1$$
$$f(1) = f(0) + g(0) = 1 + 2 = 3$$
$$f(2) = f(1) + g(1) = 3 + 4 = 7$$
$$f(3) = f(2) + g(2) = 7 + 8 = 15$$
$$\Rightarrow f(g(1)) = f(4) = 15 + 16 = 21$$

21. (d)

*$a[N] \rightarrow$ array of 'N' pointers.
These 'N' pointers point to functions:

$$*(*(*a[N])())()$$

These functions return pointers:

$$*(*(*a[N])())()$$

These pointers are pointers to function:

$$*(*(*a[N])())()$$

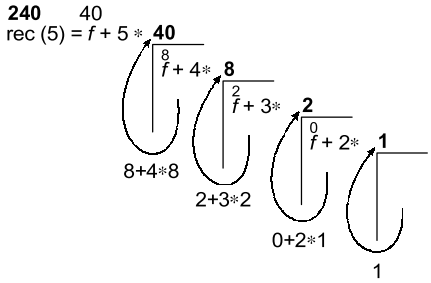
These functions return pointer to char:

$$*(*(*a[N])())()$$

22. (d)

Compiler error: Cannot modify a constant value.
P is a pointer to a "constant integer" and we are trying to change the value of the "constant integer".

23. (240)



Returning from $x = 1$ to 5, 'f' value is updated being a static variable = $40 + 5 \times 40 = 240$.

24. (9)

In function _1_*abc* ( ), 'i' value will be incremented after returning the value to main.
At the time of returning i is 10.
After returning i is 11.
But this 'i' is local to _1_*abc* ( ) and hence not reflected in main function. And in main function since it is pre-decrement operation, the changed value will be reflected.

25. (5)

$f(1, 3)$
$f(0, f(1, 2))$
$f(0, f(0, f(1, 1)))$
$f(0, f(0, f(0, f(1, 0))))$
$f(0, f(0, f(0, f(0, 1))))$
$f(0, f(0, f(0, 2)))$
$f(0, f(0, 3))$
$f(0, 4)$
$\Rightarrow 5$

26. (c)

2000

X	J	S	H	A	K	Z	A	A	O	H	E
Y	0	1	2	3	4	5	6	7	8	9	10

$x = 2000,\quad y = 2000$
$y[10] = E$ (69 in ASCII)
$y[7] = A$ (65 in ASCII)
$x + y[10] - y[7]$
$= 2000 + 69 - 65 = 2004$

Therefore it prints from the array starting at address; 2004 to the end i.e., "K Z A A O H E".