POSTAL Study Course 2018

Computer Science & IT

Objective Practice Sets

Programming and Data Structure

Contents

MADE EASY

India's Best Institute for IES, GATE & PSUs

For MADE EASY Students Only

1 CHAPTER

Computer Science & IT

Programming Methodology

Q.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.

Q.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

Q.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?

Q.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

Q.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 $k = i/8 + j * 4$;

Q.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 & 01) 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*'.

Q.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 _____.

Q.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

Q.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

Code:

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

Q.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

Q.11 Assume the following values are inserted into a BST in the given order.

4,5,7,1,2,3,6,9,8,10.

Consider the following function:

```
void tree (Node * root)
{   if (root == NULL) return;
    printf("%d", root → data);
    tree (root → left);
    tree (root → right);
    return;
}
struct node
{   int data;
    struct node * left;
    struct node * right;
} Node;
```

Find the output printed by the above function, if the root pointer of the BST is passed to the "tree" function.

(a) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
(b) 4, 1, 2, 3, 5, 7, 6, 9, 8, 10
(c) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
(d) 4, 5, 7, 1, 2, 3, 6, 9, 8, 10

Q.12 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

Q.13 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.

Q.14 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 lexical scoping?

(a) 2, 3 (b) 3, 2
(c) 2, 2 (d) 3, 3

Q.15 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)

Q.16 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) runtime error (d) compile time error

Q.17 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)); }
}
```

What is the return value of *f*(2, 10)?

Q.18 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

Q.19 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

Q.20 Which of the following is illegal statement in *C*.

(a) int (**p) [ ]; (b) int*(*p) ( );
(c) int (*f ( )) [ ]; (d) int*f( ) [ ];

Q.21 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))?

Q.22 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])( )) ( );

Q.23 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

Q.24 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).

Q.25 Find the output of the following program.

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

Q.26 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)));
}
```

Q.27 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"

Q.28 Consider the following code

```
int Do (char *gate)
{
    char *gate1 = gate;
    char *gate2 = gate + strlen (gate) – 1;
    while (gate1 < gate)
    {
        if (*gate1 ++ ! =*gate2 - -)
        return 0;
    }
    return 1;
}
```

What is the functionality of above function Do( )?
(a) Check whether string is odd palindrome
(b) Check whether the string is even palindrome
(c) Check whether the string is palindrome
(d) None of the above

Q.29 Consider an expression:

$$(j + ((i - j) \& - (i < j))).$$

Which of the following is true about the given expression, where i, j are integers?
(a) Finds the maximum of two integers i and j
(b) Finds the minimum of two integers i and j
(c) Finds the G.C.D of two integers i and j
(d) Finds the L.C.M of two integers i and j

Q.30 Assume i and j are small integers. Which of the following code snippets swaps i and j without third variable? (^ is a XOR operation bitwise).

(a)
```
i = i + j
j = i – j
i = i – j
```

(b)
```
i = i * j;
j = i / j;
i = i / j;
```

(c)
```
i = i ^ j;
j = i ^ j;
i = i ^ j;
```

(d) All of these

Q.31 Consider the following program.

```
variable l;
procedure K1 (var l)
    begin
        print (– –l );
    end
procedure K2 (var m)
    begin
        K1 (m);
    end
begin
    l = 6;
    K2 (l );
    print (l );
    l = l + 2;
    K1 (l );
end
```

If static scoping is used, which of the following is correct output for the above program?
(a) 5, 6, 7 (b) 5, 5, 6
(c) 6, 6, 8 (d) 5, 6, 8

Q.32 Consider the following C program.

```
int x ;
int main( )
{
   int y ;
   //
   //
   {
      int z;
      //
   }
}
```

Which variable has the longest scope in the above program?

(a) x
(b) y
(c) z
(d) All variables have same scope

Q.33 Choose the identical statement.

(a) (*Ptr) → element AND Ptr → element.
(b) (*Ptr) . element AND Ptr → element.
(c) *(Ptr. element) AND Ptr → element.
(d) *Ptr. element AND Ptr → element.

Q.34 For for loop:

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

prints

(a) 0101010101 (b) 0111111111
(c) 0000000000 (d) 1111111111

Q.35 Consider the following function

```
int evaluation (int n)
{   if (n <= 2)
    return 1;
    else
    return (evaluation (floor(sqrt (n))) + n);
}
```

What will be returned if n is 100 _____.

Q.36 Let m, n be positive integers. Define $Q(m, n)$ as

$Q(m, n) = 0$, if $m < n$
$Q(m - n, n) + p$, if $m \geq n$

Then $Q(m, 3)$ is (a div b, gives the quotient when a is divided by b)

(a) a constant (b) $p \times$ (m mod 3)
(c) $p \times (m \text{ div } 3)$ (d) $3 \times p$

Q.37 Consider the following function

```
void function (int * A, int n)
{
    if (n! = 0)
    {
        printf ("%d", A[n - 1]);
        function (A + 1, n - 1);
    }
}
```

Find the third output produced by the function call function (A, 5), and A is an array initially holds {10, 20, 30, 40, 50}.

Q.38 Consider the following program.

```
int i = 1;
int main ()
{
    int a[] = {0, 1, 2};
    f (a[i], i);
    printf ("%d", a[i]);
}
```

```
void f(int x, int y)
{
    y++;
    x = 5*i;
}
```

If above function $f()$ uses "call by name" technique, what is the output printed?

Q.39 Match **column-I** with **column-II**

Column-I

A. float *(*f) ( );
B. float (*f) ( );
C. float (*a) [8];
D. float *(*a) [8];

Column-II

1. a pointer to an array of 8 floats.
2. a pointer to an array of 8-pointer to floats.
3. a pointer to a function that returns float.
4. a pointer to a function that returns a pointer to a float

Codes:

	A	B	C	D
(a)	4	3	1	2
(b)	3	4	1	2
(c)	4	3	2	1
(d)	None of these			

Q.40 Which of the following is the correct output for the 'C' program given below?

```
# include < stdio.h>
int main()
{
    int arr [3] = {2, 3, 4};
    char *p;
    p = (char*) arr;
    printf ("%d", *p);
    p = p+1;
    printf ("%d \ n", *p);
    return 0;
}
```

(a) 2 3 (b) 2 0
(c) 1 0 (d) Garbage values

Q.41 Consider the following function

```
void Zigzag (int * A, int n)
{
    if (n! = 0)
    {
        printf ("%d", A[n – 1]);
        Zigzag (A + 1, n – 1);
    }
}
```

What is the 3rd output produced by the function call Zigzag (A, 5), and A is an array initially holds {10, 20, 30, 40, 50}.

Q.42 What is the output of the following C program?

```
# include < stdio.h>
int f (int*a, int n)
{   if (n == 1) return 1;
    else if (*a % 2 == 0)
    return (*a + f (a + 1, n – 1));
    else return (*a × f (a + 1, n – 1));
}
int main ( )
{   int a[ ] = {10, 5, 20, 2, 3, 1};
    printf ("%d", f (a, 4));
    return 0;
}
```

■■■■

Answers Programming Methodology

1.	(b)	2.	(a)	4.	(d)	5.	(a)	6.	(a)	8.	(d)	9.	(b)	10.	(d)	11.	(b)		
12.	(c)	13.	(c)	14.	(d)	15.	(c)	16.	(a)	18.	(b)	19.	(a)	20.	(d)	22.	(d)		
23.	(d)	27.	(c)	28.	(c)	29.	(b)	30.	(d)	31.	(a)	32.	(a)	33.	(b)	34.	(b)		
35.	(d)	37.	(c)	40.	(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*) f_2 (int *)
The correct declaration for (d) is int *(**f*) fun(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 bitwise 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.

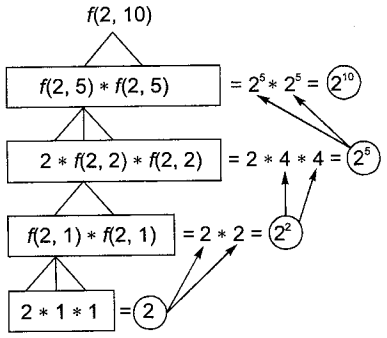
```
while (n! = 0)
{
    if (n & 01) x ++;
    /* performs bit wise AND operator and if
    condition is satisfied if result
    contains atleast one 1.
    n >> = 1
}
```

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-refrenced, 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) delcares pointer. (* P) [4] is array pointed by pointer.

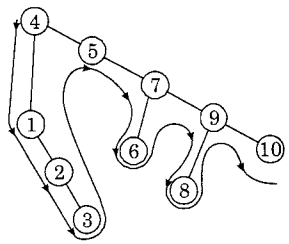
C : * P [4] declares array of pointers.

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. (b)



∴ 4,1,2,3,5,7,6,9,8,10 is printed by the function.

12. (c)

X : $b + c * 2$ is 8

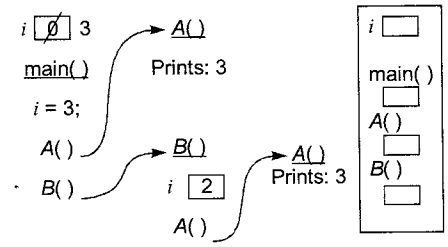
Case 8: prints 3 then default case prints 2

∴ Output prints 32.

13. (c)

Return statement can contain an expression.

14. (d)



Output printed by the code: 3, 3

15. (c)

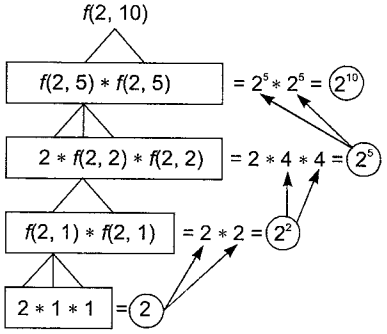
Only constants or enums can be used with cases of switch. 2*4 is a constant expression.

16. (a)

Loop repeated for x = 0, 1, 2, 3 and when x = 4, A[4] is null string, so loop will be terminated.

17. (1024)

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



18. (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

$a = 20$, $b = 30$

19. (a)

Iterations of for statement

	i = 0		i = 1		i = 2		i = 3		i = 4	
for :	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.

20. (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.

21. (21)

$f(g(1))$:

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

22. (d)

*a[N] → array of 'N' pointers.
These 'N' pointers point to functions:

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

These functions return pointers:

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

These pointers are pointers to function:

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

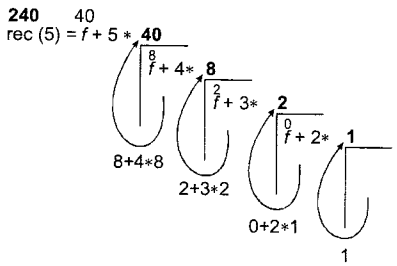
These functions return pointer to char:

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

23. (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".

24. (240)



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

25. (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.

26. (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

27. (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
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".

28. (c)

Here two pointers are used gate1 and gate2. Gate 1 pointer points to beginning and gate 2 points to the end.

Loop is set up that compares the characters pointed by these two pointers. If the characters do not match, then it's returning 0 i.e. it is not a palindrome. It returns1 for both even and odd palindrome (i.e., when match occurs for entire loop).

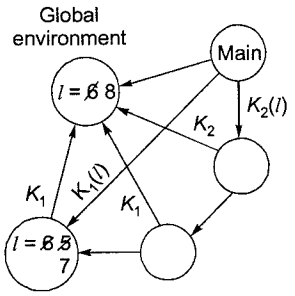
29. (b)

This code returns the minimum of two integers *i* and *j*.

30. (d)

All the three code snippets do the same work. They swap *i* and *j* without 3rd variable.

31. (a)



In static scoping the function call will search for the variable in local environment. If not found then it searches in global environment. All the functions access the global environment.

∴ 5, 6, 7 is correct output.

32. (a)

x is global variable hence accessible everywhere. *y* is limited to main and *z* is accessible to only in the block where it is declared.

Therefore, *x* has the longest scope.

33. (b)

As both the operators [→ and (*) .] are used to access the members of the structure through pointers.

34. (b)

Dynamic data structures have the capacity to grow. Hence, they need some sort of memory allocation during runtime. The heap is required to provide this memory.

35. (d)

As ++*i* will increase i from 0 to 1 before execution of *i* and 1.

36. (114)

(Evaluation (floor (sqrt (100))) + 100)

⇓

(Evaluation (floor (sqrt (10))) + (10)

⇓

(Evaluation (floor (sqrt (3))) + 3)

⇓

1

⇓

114

37. (c)

Let $m > n$. Let m/n yield a quotient x and remainder y. So, $m = n * x + y$ and $y < m$ div 3 is the quotient when m is divided by 3. So, that many times p is added, before we terminate recursion by satisfying the end condition $Q(m, n) = 0$, if $m < n$. Hence the result.

38. (50)

function (A, 5)

A → | 10 | 20 | 30 | 40 | 50 |

1st: prints "50"

function (A + 1, 4)

function (A, 4)

A → | 20 | 30 | 40 | 50 |

2nd: prints "50"

function (A + 1, 3)

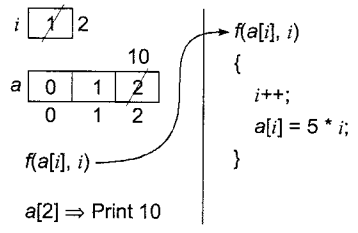
function (A, 3)

A → | 30 | 40 | 50 |

3rd: prints "50"

3rd value printed by function is : 50

39. (10)



40. (b)

	1000	1001	1002	1003	
arr →	00000010	00000000	00000011	00000000	...

as p is a char pointer on $p+1$, it will point to 1001 address.

41. (50)

First call: Zigzag (A, 5)

A → | 10 | 20 | 30 | 40 | 50 |

1st: prints "50"

Zigzag (A + 1, 4)

Second call: Zigzag (A, 4)

A → | 20 | 30 | 40 | 50 |

2nd: prints "50"

Zigzag (A + 1, 3)

Third call: Zigzag (A, 3)

A → | 30 | 40 | 50 |

3rd: prints "50"

3rd value printed by Zigzag is : 50

42. (115)

a[ ] = | 10 | 5 | 20 | 2 | 3 | 1 |

So $f(a, 4)$ is the first call

$*a + f(a + 1, 3)$ even $(*a)$

$*a \times f(a + 2, 2)$ odd $(*a)$

$*a + f(a + 3, 1)$ even $(*a)$

1 Termination

So output will be

$= 10 + [5 \times (20 + 1)]$

$= 10 + [5 \times 21]$

$= 10 + 105 = 115$

■■■■

2 CHAPTER

Arrays

Q.1 Which of the following C expressions access the $(i, j)^{th}$ entry of an (m × n) matrix stored in column major order?

(a) $n \times (i-1) + j$ (b) $m \times (j-1) + i$

(c) $m \times (n-j) + j$ (d) $n \times (m-i) + j$

Q.2 Consider 3 dimensional Array A[90] [30] [40] stored in linear array in column major order. If the base address starts at 10, what is the location of A [20] [20] [30]? Assume the first element is stored at A[1][1][1].

Q.3 Consider a 3-heap tree which is similar to 2-heap tree. Every node in 3-heap contains maximum of 3-children. If Array is used to store the element of 3-heap, find the children of node i ? Assume the first element of array is at 1.

(a) $3i$, $3i + 1$, $3i + 2$ (b) $3i - 1$, $3i$, $3i + 1$

(c) $3i + 1$, $3i + 2$, 3 (d) None of these

Q.4 In a compact single dimensional array representation for lower triangular matrices of size $n \times n$, non-zero elements of each row are stored one after another, starting from the first row, the index of the $(i, j)^{th}$ element of the lower triangular matrix in this new representation is

(a) $i + j$ (b) $i + j - 1$

(c) $j + \frac{i(i-1)}{2}$ (d) $i + \frac{j(j-1)}{2}$

Q.5 Consider the following function.

```
int search (int A[ ], int k, int l, int h)
{
    int m;
    if (l == h)
    if (k == A[l]) return l;
        else return –1;
    m = ⌊(l + h)/2⌋;
    if (k ≤ A [m])
        return search (A, k, l, m);
    else
    return search (A, k, m+l, h);
}
```

Above function is implemented to search a key in the sorted array with binary search concept. Find the index of key 15 returned by the above function, if array has the following elements and $l = 0$, $h = 8$ are passed to the function along with array and key.

A	12	14	15	15	15	18	110	120	125
	0	1	2	3	4	5	6	7	8

(a) 2 (b) 3

(c) 4 (d) None of these

Q.6 Consider a two-dimensional array with elements stored in the form of lower triangular matrix. How many elements must be crossed to read A[4, 2] from the array A[–6,..., +8, –6,..., +8] whose base address is 1000? (Assume elements are stored in row major order).

Q.7 Consider the following C code

```
int *P, A[3] = {0, 1, 2};
P = A;
*(P + 2) = 5;
P = A++;
*P = 7;
```

What are the values stored in the array A from index 0 to index 2 after execution of the above code?

(a) 7, 5, 2 (b) 7, 1, 5

(c) 0, 7, 5 (d) None of these

Q.8 Let's look about the algorithm

```
int temp, j, i;
for (i = 1; i < n; i++)
{
    temp = A[i];
```

```
for (j = i – 1; j ≥ 0 && (A([j] > temp); j – –)
A [j+1] = A[j];
A[j] = temp;
}
```

If the array is in reverse sorted order then time complexities will be

(a) $O(n)$ (b) $O(n \log_2 n)$
(c) $O(n^2)$ (d) $O(\log_2 n)$

Q.9 Suppose that we have an array of n data records to sort and that the key of each record has the value 0 or 1. An algorithm for sorting such a set of records require _____ running time.

(a) $O(1)$ (b) $O(n)$
(c) $O(n^2)$ (d) None of these

Q.10 Consider an array A has n-elements in which every element is less than $2n$. What is the running time to check whether the given array has distinct elements?

(a) $O(1)$ (b) $O(n)$
(c) $O(n \log n)$ (d) $O(n^2)$

Q.11 Given an array with both +ve and –ve numbers. Find the two elements such that their sum is closest to zero

Ex.: 60 –10 70 –80 85 gives –80 85

What is the tightest upper bound to solve this problem?

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

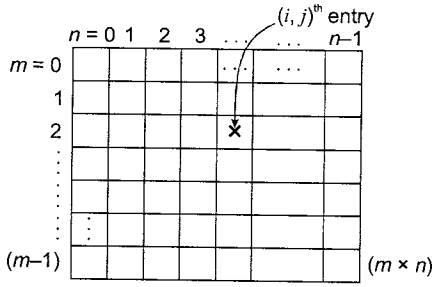
■■■■

Answers Arrays

1. (b) 3. (b) 4. (c) 5. (a) 7. (d) 8. (c) 9. (b) 10. (b) 11. (a)

Explanations Arrays

1. (b)



It is stored in column major order.

We need to cross $(j-1)$ columns, since we have 'm' elements in 1 coloum.

$\therefore$ $[m(j-1)]$ in j^{th} column $(i-1)$ elements to be crossed.

$\therefore$ $(i, j)^{th}$ location is $= m * (j-1) + i$

2. (23699)

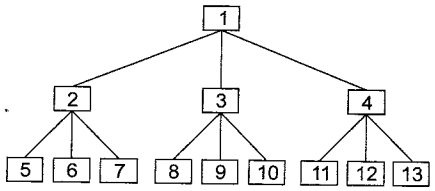
	i	j	k
Let	$A[r_1]$	$[r_2]$	$[r_3]$
	90	30	40
	(planes)	(rows)	(colums)

For column major order

loc $(A(i, j, k))$ = Base Address $+(i-1)r_2r_3+(K-1)r_2+(j-1)$

$= 10 + 19 * (30)(40) + 29 * (30) + 19 = 23699$

3. (b)



Children of 4 :

$4 * 3 - 1 = 11$

$4 * 3 = 12$

$4 * 3 + 1 = 13$

$\therefore$ $(i * 3) - 1$, $i * 3$ and $(i * 3) + 1$ are children of i.

4. (c)

The number of elements to be skipped to reach to i^{th} row $= \frac{i(i-1)}{2}$ to reach to j^{th} column $= \frac{i(i-1)}{2} + j$.

5. (a)

(ℓ)				(m)				(h)
12	14	15	15	15	18	110	120	125
(ℓ)		(m)		(h)				
12	14	15	15	15				
(ℓ)	(m)	(h)						
12	14	15						

$15 \Rightarrow l == h$ is true and $k == A[l]$.

$[l = h = 2]$

$\therefore$ Option (a) is correct.

6. (63)

The given lower triangular matrix can be represented as

	−6	−5	−4	+8
−6	a_{11}			
−5	a_{21}	a_{22}		
−4	a_{31}	a_{32}	a_{33}	
.	.	.		
.	.	.		
.	.	.		
.	.	.		
+8	a_{81}	a_{82}	. . . . .	a_{88}

Let (i, j) be the element to be accessed.

We must cross upto $(i-1)^{th}$ row.

Number of elements upto $(i-1)^{th}$ row or 10^{th} row

$= 1 + 2 + 3 + ... + [(i-1) - (l_{bi}) + 1]$

$[l_{bi} \rightarrow$ lower bound of $i]$

$= 1 + 2 + 3 + ... (3 - (-6) + 1)$

$= 1 + 2 + 3 + ... + (10)$

$$= \frac{10 \times 11}{2} = 55$$

In i^{th} row we must cross $(j - l_{bj})$ elements.

$[l_{bj} \rightarrow$ lower bound of $j]$

$= 2 - (-6) = 8$

$\therefore$ In total = 55 + 8 = 63 elements need to be crossed.

7. (d)

$P = A++$; produces compiler error.

So execution of the given code is not possible.

$A++$ asks the compiler to change the base address of an array, but compiler knows A is array hence once it is declared, compiler will not allow to change the address.

8. (c)

In this programme first for loop will run n-times and second for loop also run n-times, because our array is reverse sorted then second loop will also run and the total time complexity for reverse sorted order will be $O(n^2)$.

9. (b)

Using counting sort, it takes linear time.

10. (b)

Using counting sort, single scan will identify if there exist any repeated element in the given array. Therefore, it takes $O(n)$ time.

11. (a)

1. First sort elements $\rightarrow n \log n$.
2. Add (i) and in temp at last and before that set +ve closest = max and –ve closest-min

```
temp = A[i] + A[j]
if (temp > 0)
{
    if (temp < positive closest)
    positive closest = temp;
    j − −;
}
else if (temp < 0)
{
    if (temp > −ve closest)
    negative closest = temp;
    i++;
}
```

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

■■■■

3 CHAPTER

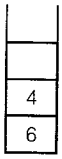
Stack

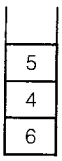
Q.1 Computes the postfix equivalent of following expression

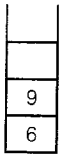
$$3*\log(x+1)-\frac{a}{2}$$

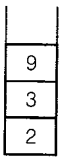
(a) 3x1 + log*a2/- (b) 31x + log * a2/-
(c) 31x log + *a2/- (d) 31 logx + * a2/-

Q.2 In evaluating the arithmetic expression 2 * 3 – (4 + 5), using stack to evaluate.
Which of the following is stack configuration is not possible

(a) 

(b) 

(c) 

(d) 

Q.3 Consider the following elements;

1, 2, 3, 4, 5, 6, 7

Use the Binary Max heap to construct a tree with above elements from 1 to 7. Find the children of root.

(a) 5, 6 (b) 4, 6
(c) 3, 6 (d) 2, 6

Q.4 A queue is implemented using two stacks *A* and *B*. Consider the following code.

```
void enqueue (int value)
{
    while (!B.isEmpty ( ))
    A.Push (B.Pop ( ));
    A.Push (value);
}
int dequeue( )
{   while (! A.isEmpty ( ))
    {X}
    return B.Pop( );
}
```

If enqueue is implemented using two stacks *A* and *B* with operations Push, Pop and isEmpty of stack then find the missing statement *X* to implement dequeue operation of queue.

(a) A.Push (B.Pop( ));
(b) B.Push (A.Pop( ));
(c) A.Pop (B.Push( ));
(d) B.Pop (A.Push( ));

Q.5 Consider the following code. Assume initially stack is empty.

```
ch = getchar( );
while (ch != '\n')
{  if ((ch != '+') && (ch != '–') && (ch != '*') &&
   (ch != '/'))
   {  E = atoi (ch); /*converts character to
      integer*/
      push (E);
   }
   else
   {  x1 = pop( );
      x2 = pop( );
      switch (ch)
      {  case '+' : E = x1 + x2; break;
         case '–' : E = x1 – x2; break;
         case '*' : E = x1 * x2; break;
         case '/' : E = x1 / x2; break;
         default : printf ("Not a valid character");
         break;
      }
      push (E);
   }
   ch = getchar ( );
}
```

Assume +, –, * and / operators are used in the expression, the variable ch can hold either operator or integer, x_1 and x_2 are integer variables. Find the content of the stack after executing the program, if the input is combination of operators and integers?

(a) Value of infix expression
(b) Value of prefix expression
(c) Value of postfix expression
(d) None of these

Q.6 Consider the following infix expression of C.

$g + a - t + e * e / x * a / m - r + e - s + u - l + t$

Find the length of substring in the equivalent postfix expression for the above infix and that substring has only operators, and the number of distinct operators and the length of substring is maximum compared to any other substring.

Q.7 Find postfix expression for the following infix expression? Assume ↑ as the highest precedence and follows right to left associativity.

Infix: $(a + b) \uparrow (p + q) \uparrow (r * s * t)$

(a) $ab + pq + \uparrow rs * t * \uparrow$
(b) $ab + pq + \uparrow\uparrow rs * t *$
(c) $ab + pq + rs * t * \uparrow\uparrow$
(d) $ab + pq + rst * * \uparrow\uparrow$

Q.8 Assume ↑ is power operator and it has the highest precedence and follows right associativity. What is the output of the following postfix expression evaluated using operand stack?

(*Note:* 2 ↑ 3 = 8)

9 9 1 * 9 1 ↑ / + 9 – 9 +

(a) 10 (b) 19
(c) 18 (d) 81

Q.9 Consider the implementation of multiple stacks in single array S of size P from index 0 to $P-1$. Size of each stack is Q. The following function PUSH ( ), used to push data x on to a particular stack i where T_i is used as top variable for stack i (i indicates stack number).

```
PUSH (S, P, Q, T_i, x)
{
    if (_____A_____)
    {
        printf ("stack overflow");
        exit (1);
    }
    else
        T_i++;
    S[T_i] = x;
}
```

Stack 0 stores elements from 0 to $Q-1$, stack 1 stores from q to $2Q-1$, and similarly other stack will store elements. Which of the following is the correct expression to replace A in the above function?

(a) $T_i == \left(\frac{P}{Q} \times i - 1\right)$

(b) $T_i == \left(\frac{P}{Q} \times i + 1\right)$

(c) $T_i == \left(\frac{P}{Q} \times (i-1) - 1\right)$

(d) $T_i == \left(\frac{P}{Q} \times (i+1) - 1\right)$

Q.10 Assume stack A has the entries p, q and r (with p on top and r on bottom). Initially stack B is empty. An entry popped out of stack A can be printed immediately or pushed to stack B. A entry popped out of stack B can only be printed.

What is the least number of stack permutations of input sequence that start with a particular letter?

Q.11 The post fix form of $A \$ B * C - D + E / F / (G + H)$ is

(a) $AB \$ C * D - EF / GH + / +$
(b) $AB \$ * C - D + EF / GH / +$
(c) $AB \$ C + D - EF / GH - / +$
(d) $AB \$ C - D * EF / GH / ++$

Q.12 The prefix form of $A - B / (C * D \$ E)$ is

(a) $- / * AB * (C - D)$ (b) $- ABCD * DE \$$
(c) $-A / B * C \$ DE$ (d) $- A / BC * \$ DE$

Q.13 Assume ↑ is power operator and it has the highest precedence and follows right associativity. What is the output of the following postfix expression evaluated using operand stack?
(Note: 2 ↑ 3 = 8)

7 7 1 * 7 1 ↑ / + 7 – 7 +

Q.14 Let S be a stack of size $4 \geq 1$ and it is initially empty. Suppose we push the numbers 1, 2, 3, 4 in sequence and then perform 4 pop operations. Let one push operation takes 5 ns; one pop operation takes 5 ns; the time between the end of one such stack operation and the start of the next operation is 2 ns. The stack-life of a particular element $p \geq 1$ is defined as the time elapsed from the end of push (p) to the start of the pop operation that removes p from the stack. The average stack-life of an element of this stack is_____ (in ns).

Q.15 Consider the following infix expression which is to be converted to postfix expression using stack.

$(((P + Q) * (R + S)) / T) + (A * (B + C))$

The sum of all unique possible heights of stack when converting from infix to postfix is _____.

■■■■

Answers Stack

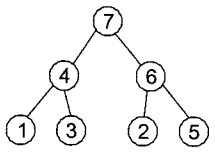
1. (a) 2. (d) 3. (b) 4. (b) 5. (c) 7. (c) 9. (d) 11. (a) 12. (c)

Explanations Stack

1. (a)
Postfix equivalent of $3 * \log(x + 1) - a / 2$
$= 3 * \log(x1 +) - a2 /$
$= 3 \log(x1 +) * a2 /-$
$= 3\, x1 + \log * a2 /-$

2. (d)
For evaluating this using stack, starting from the left, we have to scan one by one, if it is an operand, push. If an operator encountered pop twice, apply operator on the popped out entries and push the result onto the stack if we follow this, we can find configuration in option (d) is not possible.

3. (b)



4 and 6 are children of root 7.

4. (b)
Dequeue is implemented as follows:
1. If Stack A has elements then Pop from Stack A and Push to Stack B one-by-one until Stack A is empty. Otherwise simply follow next step. B.Push (A.Pop( ));
2. Now, top of Stack B holds the first element of queue hence return the top by deleting it. B.Pop( );

5. (c)
Given postfix expression is evaluated using the stack.

6. (2)
$ga + t - ee * x / a * m /+ r - e + s - u + l - t +$
∴ Substring: /+ has only operators and two distinct operators.
Any other substring with only operators contains only one length.
∴ 2 is length of substring.

7. (c)

The given expression is:

$(a + b)\uparrow (p + q) \uparrow (r * s * t)$

$= (ab+) \uparrow (pq+) \uparrow ((rs*) * t)$

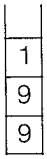
$= (ab +)\uparrow (pq+) \uparrow ((rs*) * t)$

$= (ab+) \uparrow (pq + rs * t*)\uparrow$

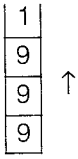
$= (ab + pq + rs* t*) \uparrow\uparrow$

$= ab + pq + rs * t * \uparrow\uparrow$

8. (10)

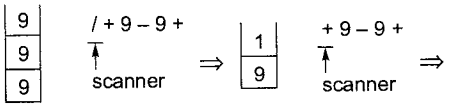


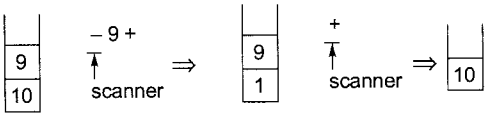
Push the operands one by one when operator comes, POP the top two operands, evaluate the value and then push result on to the stack.



$\uparrow / + 9 - 9 +$

Scanner scans power operator then pop 1 and 9, evaluate 9^1 and then push result back on the stack.

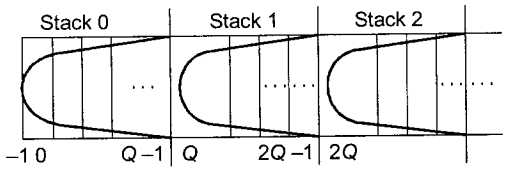




9. (d)

While performing push operation on to stack we always perform "Overflow condition check", to ensure if stack is full or not.

Number of stacks possible is P / Q.



The initial value of stack pointer $T_{i+1} = \frac{P}{Q} * i - 1$

First element of next stack pointer

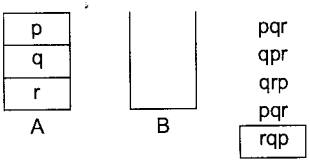
$$T_{i+1} = \frac{P}{Q}(i-1)$$

$\therefore$ Overflow condition for stack i is

$$(T_i = \frac{P}{Q}(i+1) - 1)$$

10. (1)

The following are possible stack permutations



Total stack permutations possible with the given sequence is 5. Here the least number of permutations starting with particular letter is 1 (Starting with r).

11. (a)

$A \$ B * C - D + E / F / (G + H)$

	Symbol	Postfix expression	Stack top
(1)	A	A	∞
(2)	$	A	$
(3)	B	AB	$
(4)	$*$	$AB\$$	$*
(5)	C	$AB\$C$	$*
(6)	$-$	$AB\$C*$	$-
(7)	D	$AB\$C*D$	$-
(8)	$+$	$AB\$C*D-$	$+
(9)	E	$AB\$C*D-E$	$+
(10)	$/$	$AB\$C*D-E$	$+ /
(11)	F	$AB\$C*D-EF$	$+ /
(12)	$/$	$AB\$C*D-EF/$	$+ /
(13)	$($	$AB\$C*D-EF/$	$+ / (
(14)	G	$AB\$C*D-EF/G$	$+ / (
(15)	$+$	$AB\$C*D-EF/G$	$+ / (+
(16)	H	$AB\$C*D-EF/GH$	$+ / (+
(17)	$)$	$AB\$C*D-EF/GH+/+$	

Hence (a) is the correct option.

12. (c)

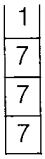
$A - B / (C * D\$E)$

	Symbol	Postfix expression	Stack top
(1)	)		)
(2)	E	E	)
(3)	$	E	$)
(4)	D	DE	$)
(5)	$*$	$\$DE$	$*)
(6)	C	$C\$DE$	$*)
(7)	(	$*C\$DE$	
(8)	$/$	$*C\$DE$	/
(9)	B	$B*C\$DE$	/
(10	)	$-$	$/B*C\$DE-$
(11)	A	$-A/B*C\$DE$	

Hence option (c) is correct.

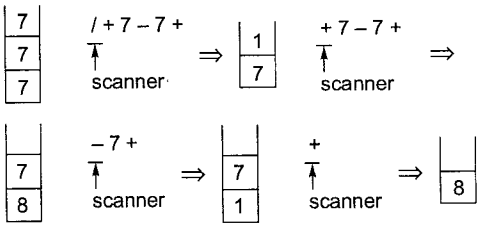
13. (8)

Push the operands one by one when operator comes, POP the top two operands, evaluate the value and then push result on to the stack.

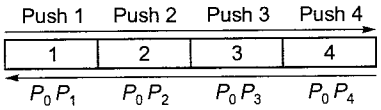


↑ / + 7 – 7 +

Scanner scans power operator then pop 1 and 7, evaluate 7^1 and then push result back on the stack.



14. (23)



Time for 1 stack operation = 5 ns

Total number of push = 4

Total number of pop = 4

Stack life of element 1

$= 5\text{ ns} \times 3 + 5\text{ ns} \times 3 + 7 \times 2\text{ ns}$

$= 15\text{ ns} + 15\text{ ns} + 14\text{ ns} = 44\text{ ns}$

Stack life of element 2

$= 5\text{ ns} \times (2 \times 2) + 5 \times 2\text{ ns}$

$= 20\text{ ns} + 10\text{ ns} = 30\text{ ns}$

Stack life element 3

$= 5\text{ ns} \times (1 + 1) + 3 \times 2\text{ ns}$

$= 10\text{ ns} + 6\text{ ns} = 16\text{ ns}$

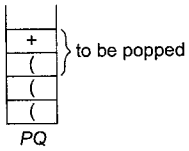
Stack life of element 4

$= 5\text{ ns} \times (0 + 0) + 2\text{ ns} \times 1 = 2\text{ ns}$

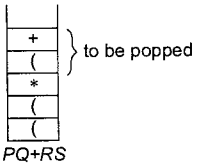
∴ Average stack life

$$= \frac{44+30+16+2}{4} = 23\text{ ns}$$

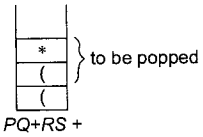
15. (15)



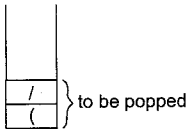
Now ') ' comes in Infix exp, pop up to first occurrence of ' ('



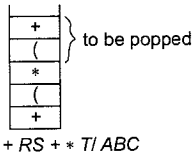
Now ') ' comes in Infix expression pop upto occurrence of ' ('.



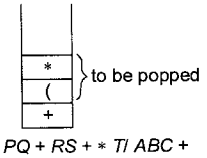
Again ') ' comes pop till first ' (' comes



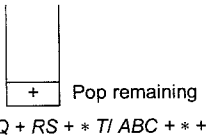
Now ') ' comes pop all elements



Now, ') ' comes, pop upto ' ('



Now ') ' comes pop until ' (' comes maximum possible stack height is 5



$PQ+RS+*\ T/ABC+* \Rightarrow PQ+RS+*T/ABC+*+$ in the postfix expression.

∴ The sum of all unique possible heights

$= 1 + 2 + 3 + 4 + 5 = 15$

4 CHAPTER

Queue

Q.1 In breadth first search of graph, which data structure is used.

(a) Stack (b) Queue
(c) Array (d) Linked List

Q.2 If the number 4, 3, 2, 1 are placed in queue (in that order), and then removed one at a time in what order they will be removed?

Q.3 Let q be a queue and S be a stack. Assume that q and S are initially empty.

```
enqueue (q, 8);
enqueue (q, 3);
Push (S, 7);
Push (S, 9);
for (i = 0; i < 5; i++)
{
    printf ("%d", dequeue (q));
    printf ("%d", Pop (S));
    enqueue (q, i);
    Push (S, i + 5);
}
```

The last integer value printed by the following code is _____.

Q.4 Suppose the number 1, 2, 3, 4, 5 and 6 arrive in an input stream in that order. Which of the following sequences can be realized as the output of double ended queue?

(*i*) 1 2 3 4 5 6 (*ii*) 2 4 3 6 5 1
(*iii*) 6 5 4 3 2 1 (*iv*) 1 5 2 4 3 6

(a) (*i*),(*iv*) only (b) (*ii*), (*iii*) only
(c) (*i*) and (*ii*) only (d) All of these

Q.5 Consider the following function.

```
void madeeasy (int n)
{
    enqueue (Q, 0);
    enqueue (Q, 1);
    for (i = 0; i < n; i++)
    {
        x = dequeue (Q);
        y = dequeue (Q);
        enqueue (Q, y);
        enqueue (Q, x + y);
        print (x);
    }
}
```

What is the functionality of above function madeeasy?

(a) Prints numbers from 0 to n – 1
(b) Prints numbers from n – 1 to 0
(c) Prints first n fibbonacci numbers
(d) Prints first n fibbonacci numbers in reverse order

Q.6 Consider the following function.

```
find (Q)
{
    while (! Q. isEmpty ( ))
        S. push (Q. dequeue ( ));
    while (! S. isEmpty ( ))
        Q.enqueue (S. pop ( ));
}
```

Assume S is stack which is initially empty and Q is queue which contain n-elements. The functions:

(1) isEmpty( ) returns true if stack/queue is empty otherwise returns false.
(2) Push( ) and Pop( ) functions are standard stack operations.
(3) enqueue( ) and dequeue( ) are queue operations.

What will be the contents of Q after execution of find (Q)? (Assume S and Q are global variables)

(a) No change to the contents of Q
(b) Q will be empty (No element in the queue)
(c) Q will be reversed (All elements are in the reverse order)
(d) None of these

Q.7 A circular array based queue q is capable of holding 7 elements. After execution of the following code, find the element at index '1', if the array is initially empty and array has indices from 0 to 6.

```
for (x = 1; x <= 6; x++)
    q.enqueue (x);
for (x = 1; x <= 3; x++)
{   q.dequeue ( );
    q.enqueue (q.dequeue ( ));
}
```

Assume enqueue and dequeue are circular queue operations for insertion and deletion respectively.

Q.8 Let q be a queue and S be a stack. Assume that q and S are initially empty. What is the last integer value printed by the following code?

```
enqueue (q, 8);
enqueue (q, 3);
Push (S, 7);
Push (S, 9);
for (i = 0; i < 5; i++)
{
    printf ("%d", dequeue (q));
    printf ("%d", Pop (S));
    enqueue (q, i);
    Push (S, i + 5);
}
```

Q.9 Suppose that stacks and queues are provided as opaque data types, offering only operations to add elements, to remove elements, and to test for emptiness. Suppose that a programmer wants to count the number of elements in a given stack or queue C, which is currently in some state t, using only one auxiliary stack or queue D. The structures C and D can be used in any way possible based on the methods they offer, but C must be restored to its state t after counting its elements. Counting elements as described above is possible for which of the following data types?

1. C is a queue and D is a queue.
2. C is a stack and D is a stack.
3. C is a queue and D is a stack.

(a) None (b) 1 and 2 only
(c) 1 and 3 only (d) 1, 2, and 3

Q.10 Suppose you are given an implementation of a queue of integers the operations that can be performed on the queue is

1. empty (Q)- returns true if the queue empty, false characters
2. delete (Q) - deletes the element at the front of the queue and return its value.
3. insert (Q, i) - insert the integer i at the rear of queue.

Consider the following function

```
void f (queue Q)
{   int i;
    if (! isempty (Q))
    {   i = delete (Q);
        f(Q);
        insert (Q, i);
    }
}
```

What operation is performed by the above function f?

(a) Leaves the queue Q unchanged
(b) Reverse the order of element in the queue
(c) Deletes the element at the front of the queue Q and inserts it at the rear keeping the other elements in the same order
(d) Empties the queue (Q)

Q.11 Consider the following code :

```
void find( )
{
    tail = head;
    struct node * x = head;
    struct node * y;
    head = NULL;
    while(x! = NULL)
```

```
{
    y = x → next;
    x → next = head;
    head = x;
    x = y;
}
}
```

Assume a queue is implemented using linked list with head and tail pointers. If head points to the first element in queue and tail points to the recently inserted element. Find the functionality of "find" if head and tail are global variables.

(a) It traverses the queue
(b) It reverses the queue
(c) It searches a node in the queue
(d) It will keep the queue as same

Q.12 In an empty queue, rear and front can be initialized as
(a) rear = front = –1
(b) rear = front = 1
(c) rear = 0, front = –1
(d) rear = –1, front = 0

■■■■

Answers Queue

1. (b) 4. (d) 5. (c) 6. (c) 9. (d) 10. (b) 11. (b) 12. (a)

Explanations Queue

1. **(b)**
Queue is suitable structure for BFS.

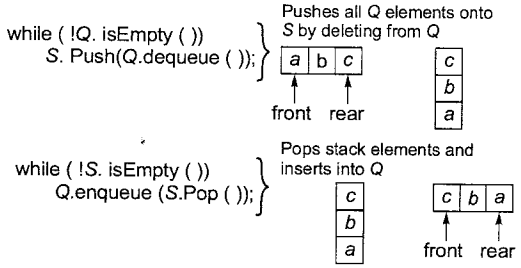
2. **(4321)**
Queue follows FIFO structure

3. **(8)**
The output sequence printed by the given code is: 8, 9, 3, 5, 0, 6, 1, 7, 2, 8
Last output is : 8

4. **(d)**
In double ended queue elements can be added and deleted from both the ends.

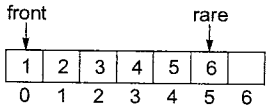
5. **(c)**
The function prints first n fibbonacci numbers. Note that 0 and 1 are initially there in the queue. This is the initial condition, for fibonacci series. In every iteration of loop, sum of two queue items is enqueued and the front item is dequeued i.e., sum of previous 2 numbers as in fibonacci series.

6. **(c)**



∴ Content of queue (Q) will be reversed after the execution of find (Q).

7. **(6)**



[After first **for** loop execution]

$x = 1$

- q.dequeue ( ); 1 will be deleted from location 0.
  q.enqueue (q.dequeue ( )); 2 will be deleted and inserted at location 6.

  $x = 2$

- 3 will be deleted from location 2.
  4 will be inserted at location 0 [circular queue]
  $x = 3$
- 5 will be deleted.
  6 will be inserted at location 1

4	6					2
0	1	2	3	4	5	6

$\therefore$ At index 1, the value 6 is present

8. (8)
The output sequence printed by the given code is: 8, 9, 3, 5, 0, 6, 1, 7, 2, 8
Last output is : 8

9. (d)
Since we are provided only one auxillary stack/ queue *D*, if the *C* is queue then we can delete elements from front of *C* and insert into *D*, and increase the count. Similarly, the same can be done if *C* is stack, i.e. pop from one stack and push into another stack while increasing the count.

10. (b)
In this program if statement execute only queue is not empty. This function is called recursively until queue becomes empty after it. Flow of control is transferred to next statement and insert the element in the reverse of the order of elements in the queue.

11. (b)
Given function reverse the queue.

```
y = x → next;
x → next = head;
head = x ;
x = y;
```

Above four statements will reverse the linked list. Head will point at tail and tail will point at head after reverse.

12. (a)
In an empty queue rear = front = –1

■■■■

5 CHAPTER

Linked Lists

Q.1 The concatenation of two lists is performed in $O(1)$ time. Which of the following implementations of a list should be used?

(a) singly linked list
(b) doubly linked list
(c) circular doubly linked list
(d) array implementation of list

Q.2 Which data structure would be most appropriate to implement a collection of values with the following three characteristics?

- Items are retrieved and removed from the collection in FIFO order.
- There is no prior limit on the number of items in the collection.
- The size of an item is large relative to the storage required for a memory address.

(a) Singly-linked list, with head and tail pointers
(b) Doubly-linked list, with only a head pointer
(c) Binary tree
(d) Hash table

Q.3 Assume given a non-empty binary search tree. If its root is passed to the following function then find the modified binary search tree?

```
void find (struct node *head)
{
    struct node * x;
    x = head;
    while (x → left ! = NULL)
    {
      x = x → left;
      if (x → left = = NULL);
      {head → data = x → data;}
    }
}
```

(a) It modifies the root data with the left node data
(b) It modifies the root data with the random left node data
(c) It modifies the root data with minimum value of tree.
(d) It modifies the root data with maximum value of tree.

Q.4 Consider the following code which is used to detect the loop in the linked list. Find the missing statement *A*?

```
p = head;
q = head → next;
while (A)
{
    if(p == q) exit (0); //loop detected
    p = p → next;
    q = (q → next)? (q→next → next): q → next;
}
```

(a) p! = NULL
(b) q! = NULL
(c) (p! = NULL) && (q ! = NULL)
(d) (p! = NULL)|| (q ! = NULL)

Q.5 Consider the following code

```
Node *find (Node * head)
{
   Node *P1 = head, *P2 = head;
   while (P2)
   {
      P1 = P1 → next;
      P2=(P2→next)? P2 → next → next: NULL;
   }
   printf ("%d", P1 → value);
}
```

Assume Node is the structure type with two members: 'value' and 'next'. Identify the node value printed by the above code if non-empty linked list header is passed to the function 'find'.

(a) First element of the list [i.e., value of the first node]
(b) Second element of the list
(c) Middle element of the list
(d) Last element of the list

Q.6 The function **delete (head, element)** is used to delete a node from the linked list by finding the node value with a given element. The parameter head is the first node of the list. Find the missing statements *A* and *B* in the following "delete" function to delete the node? (Assume all elements are distinct in the list and the function returns pointers that points to the first node of the list).

```
Node delete (Node head, int element)
{
    Node x = head;
    if (x.data == element)   return head.next;
    while (x.next ! = NULL)
    {
        if   (_____A_____)
        {
            _____B_____;
            return head;
        }
        x = x.next;
    }
}
```

(a) *A* : *x*.data == element
    B : *x*.next = *x*.next.next
(b) *A* : *x*.next.data == element
    B : *x*.next = *x*.next.next
(c) *A* : *x*.data == element
    B : *x*.next.next = *x*.next
(d) *A* : *x*.next.data == element
    B : *x*.next.next = *x*.next

Q.7 Identify the functionality of the following code when function returns the value 1. ["head" pointer point to the first node of the non-empty linked list]

```
int find (Node *head)
{
    Node *P1 = head, *P2 = head;
    if (head → next != NULL)
    {
        P1 = head → next;
        P2 = (head → next)? head →next → next:
        NULL;
    }
    while ((P1! = NULL) && (P2! = NULL))
    {
        if (P1 == P2)  return 1;
        P1 = P1 → next;
        P2 = (P2 → next != NULL)? P2 → next →
        next: NULL;
    }
    return 0;
}
```

(a) It finds the duplicate element in the list
(b) It finds the middle node in the list
(c) It finds the cycle in the list
(d) It finds the last node in the list

Q.8 Let 'new' and 'current' be two pointers. 'new' is pointing to the new node and 'current' is pointing to the some node in the doubly linked list. If new node needs to be inserted after the current node then find which of the following operations are correct?

(a)
```
new → next = current → next;
new = current → next;
new → prev = current;
(current → next) → prev = new;
```
(b)
```
new → next = current → next;
new → next = current;
new → prev = current;
(new → next) → prev = new;
```
(c)
```
new → next = current → next;
current → next = new;
new → prev = current;
(new → next) → prev = new;
```
(d)
```
new → next = current → next;
current → next = new;
new → prev = current;
(current → next) → prev = new;
```

Q.9 Struct void find (Struct node ** head)

```
{  Struct node*X = *head;
   Struct node* Y;
   Struct node * Z = NULL;
   while (X != NULL)
   {
      Y = X → next ;
      X → next = Z ;
      Z = X ;
      X = Y ;
   }
   *head = Z ;
}
```

If head is a pointer to a pointer to the first node of the list and it is passed to the 'find' function then find the list after executing the function.

(a) It adds a new node at the first
(b) It adds a new node at the last
(c) It keeps the list as same
(d) It reverses the list

Q.10 Consider the following structure for creating nodes of a double linked list.

```
struct node
{
    int data;
    struct node * x1 ; /* left pointer */
    struct node * y1 ; /* right pointer */
};
```

Let us assume P be the node in the existing linked list. Which of the following is true about the following code C_1?

Code C_1 :
$$(P \to x_1) \to y_1 = P \to y_1;$$
$$(P \to y_1) \to x_1 = P \to x_1;$$

Assume memory is made free automatically by the compiler after deletion.

(a) It deletes the node to the right of P
(b) It deletes the node to the left of P
(c) It deletes the node P from the linked list
(d) None of the above

Q.11 Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node X from given linked list?

(a) Possible if size of linked list is even
(b) Possible if size of linked list is odd
(c) Possible if X is not first node
(d) Possible if X is not last node: (a) copy the data of next of X to X (b) delete next of X.

Q.12 In a circularly linked list organization, insertion of a record involves the modification of

(a) no pointer
(b) 1 pointer
(c) 2 pointers
(d) 3 pointers

Q.13 The following C function takes single linked list of integers as parameter and rearrange the elements of the list. The function is called with the list containing integers 10, 20, 30, 40, 50, 60, 70 in given order and generate output as 20, 10, 40, 30, 60, 50, 70 after completion. What will be the correct options files so that we get desired output?

```
struct node
{
   int val;
   struct node * next;
}
void Altswap(struct node * list)
{
   struct node *p, *q;
   int temp;
   if (! list || list → next) return;
   p = list; q = list → next;
   while (q)
   {
      W
      X
      Y
      Z
      A
   }
}
```

(a) $W : p \rightarrow \text{val} = q - \text{val};$
$X : q \rightarrow \text{val} = \text{temp};$
$Y : p = q \rightarrow \text{next};$
$Z : \text{temp} = p \rightarrow \text{val};$
$A : q = p?p \rightarrow \text{next} : 0;$

(b) $W : \text{temp} = p \rightarrow \text{val};$
$X : q \rightarrow \text{val} = \text{temp};$
$Y : p \rightarrow \text{val} = q \rightarrow \text{val};$
$Z : q = p?p \rightarrow \text{next}: 0;$
$A : p = q \rightarrow \text{next};$

(c) $W : q = p?p \rightarrow \text{next}: 0;$
$X : \text{temp} = p \rightarrow \text{val};$
$Y : p \rightarrow \text{val} = q \rightarrow \text{val};$
$Z : q \rightarrow \text{val} = \text{temp};$
$A : p = q \rightarrow \text{next};$

(d) $W : \text{temp} = p \rightarrow \text{val};$
$X : p \rightarrow \text{val} = q \rightarrow \text{val};$
$Y : q \rightarrow \text{val} = \text{temp};$
$Z : p = q \rightarrow \text{next};$
$A : q = p?p \rightarrow \text{next} : 0;$

■■■■

Answers Linked List

1.	(c)	2.	(a)	3.	(c)	4.	(c)	5.	(c)	6.	(b)	7.	(c)	8.	(c)	9.	(d)
10.	(c)	11.	(d)	12.	(c)	13.	(d)										

Explanations Linked List

1. **(c)**
The concatenation of two lists is to performed on $O(1)$ time by the use of circular doubly linked list. The reason behind this, that it has the pointer in the both direction in front and backward.

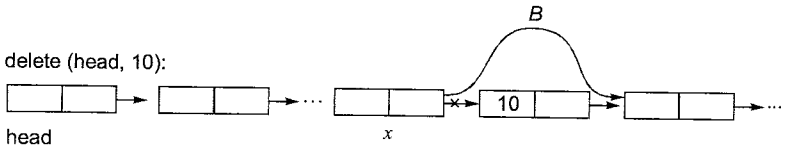
2. **(a)**
For FIFO
(i) Elements will be inserted using tail ointer. Elements will be deleted using head pointer.
(ii) In a single linked list, any number of elements ca be inserted without declaring prior size.
(iii) An item of larger size can be stored in a linked list using structures.

3. **(c)**
$x = x \rightarrow$ left is iterated until $x \rightarrow$ left ! = NULL. It finds the extreme left node to the root. This node always contain minimum value of binary search tree.
head $\rightarrow$ data = $x \rightarrow$ data will replace the root with minimum value.

4. **(c)**
Both p and q are not NULL then there is a loop in the linked list and it will be detected by slow pointer (p) and fast pointer (q).
$\therefore$ · $A : p$!= NULL && q != NULL.

5. **(c)**
$P1$ traverses node by node
$P2$ traverses by skipping one node. The node pointed by $P1$ is the middle node in the linked list when $P2$ reaches to NULL.
$\therefore$ Middle element of the list is printed by $P1 \rightarrow$ value.

6. (b)




```
if ( x.next.data == 10 )        // A
{
    x.next = x.next.next;       // B
    return head;
}
```

7. (c)

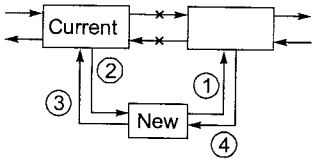
$P1$ is slower pointer, which moves to the next node in every iteration of while loop.

$P2$ is faster pointer, which moves to the next to next node in every iteration.

If there exist a cyle in the linked list (non-empty), it returns 1 by checking $P1 == P2$.

otherwise it returns 0.

8. (c)



(1) new → next = current → next;

(2) current → next = new;

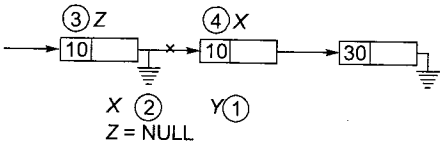
(3) new → prev = current;

(4) (new → next) → prev = new;

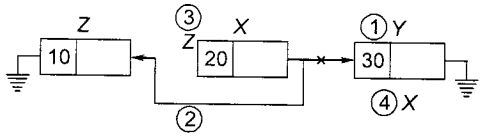
9. (d)

Assume initially X is pointing to the first node.

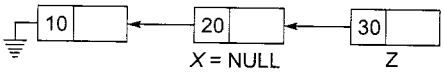
1. $Y = X \rightarrow$ next;
2. $X \rightarrow$ next = Z;
3. $Z = X$;
4. $X = Y$;



In the first iteration of loop, list is modified as above. In the second iteration of the loop, second node next is the first, which is shown below.



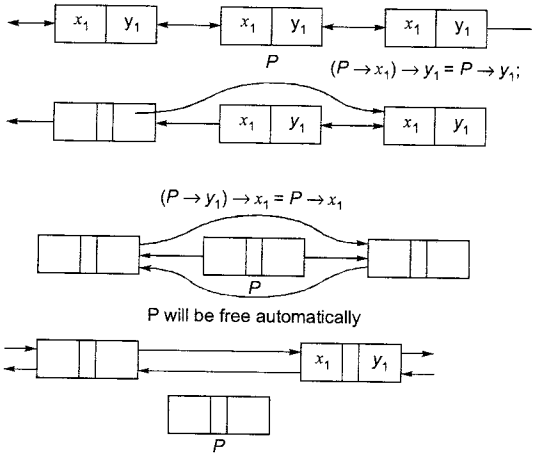
Similarly after the third iteration, 3rd node next is the second node. After the third iteration the list is reversed as following.



While loop exit due to X = NULL and finally executes *head = Z, so head will be double pointer to the node 30.

∴ list is reversed

10. (c)



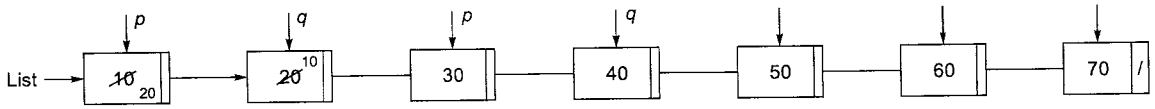
11. (d)

Struct node * temp = $x \rightarrow$ next;
$x \rightarrow$ data = temp $\rightarrow$ data;
$x \rightarrow$ next = temp $\rightarrow$ next;
free (temp);

12. (c)

For insertion/deletion of a record in circularly linked list involves 2 pointer.
But for reversing the linked list, it involves 3 pointer.

13. (d)



1. temp = 10 ;
2. $p \rightarrow$ value = $q \rightarrow$ value;
3. $q \rightarrow$ value = temp;
4. $p = q \rightarrow$ next;
5. $q = p\ ?\ p \rightarrow$ next: 0 ;

So option (d) is correct.

■■■■

6 CHAPTER

Trees

Q.1 T is a search tree of order K, its size is N and its height is H. The computation time needed to insert/delete a data item on T is

(a) $O(\log H)$ (b) $O(N)$
(c) $O(K)$ (d) $O(H)$

Q.2 Which is efficient tree structure, considering space and time complexities?

(a) AVL Tree
(b) Full Binary Tree
(c) Complete Binary Tree
(d) Binary search tree

Q.3 The inorder traversal of some binary tree produced the sequence MFIEJGZ, and post order sequence is HIFJZGE the number of nodes in left subtree of a given tree _____?

Q.4 The degree of a node in a tree is the number of children the node has if a tree has n_1 nodes of degree 1, n_2 nodes of degree 2, . . . n_m nodes of degree m, then number of leaves in the tree in terms of $n_1, n_2, \ldots n_m$

(a) $1 * n_1 + 2 * n_2 \ldots (m-1)n_m$
(b) $n_1 + n_2 + \ldots + n_m$
(c) $(n_m + n_1) * m / 2$
(d) $1 + [1 * n_2 (2-1) + 2 * n_3 \ldots + (m-1) * n_m]$

Q.5 A 2-3 tree is a tree such that (a) all internal nodes have either 2 or 3 children (b) all paths from root to the leaves have same length.

The number of internal nodes of a 2-3 tree having a leaves could be

(*i*) 4 (*ii*) 5
(*iii*) 6 (*iv*) 7

(a) (*i*) and (*iii*) (b) (*i*) and (*iv*)
(c) (*ii*) and (*iii*) (d) (*ii*) and (*iv*)

Q.6 A binary search tree is constructed by inserting the key values 1, 2, 3, 4, 5, 6, 7 in some order specified by a permutation of 1,..., 7 into an initially empty tree. Which of these permutation will lead to a complete binary search tree?

(a) 1, 2, 3, 4, 5, 6, 7 (b) 4, 6, 5, 4, 1, 2, 3
(c) 4, 2, 5, 1, 3, 5, 7 (d) 4, 1, 5, 3, 6, 2, 7

Q.7 Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. Which of the following sequence could not be the sequence of nodes examined?

(a) 2, 252, 401, 398, 330, 344, 397, 368
(b) 924, 220, 911, 244, 898, 258, 362, 363
(c) 925, 202, 911, 240, 912, 245, 363
(d) 2, 399, 387, 219, 266, 382, 381, 278, 278, 363

Q.8 Which of the following statements is/are true.

(*i*) Suppose the search for key 'k' in a binary search tree ends up in a leaf. Consider three sets A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path then any must satisfy $a \le b \le c$.

(*ii*) Operation of deletion is cumulative in the sense that deleting x and they y from a binary search tree leaves the same tree as deleting y and then x.

(a) only (*i*) (b) only (*ii*)
(c) Both true (d) Both false

Q.9 Consider three keys, k_1, k_2, k_3 such that $k_1 < k_2 < k_3$. A binary search tree is constructed with these three keys. Depending on the order in which the keys are instered, number of binary search trees possible are

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

Q.10 An AVL tree is constructed by inserting the key values 1, 2, 3, 4, 5 in some order specified by a permutation of 1, 2, 3, 4, 5 into an initially empty tree. For which of the following permutation there is no need to do any rotation at any stage during the insertion.

(a) 1, 3, 2, 5, 4 (b) 4, 2, 5, 1, 3
(c) 5, 3, 2, 1, 4 (d) 2, 3, 4, 5, 1

Q.11 Which of the following traversal is sufficient to construct Binary search tree from given traversal?

I. Preorder
I. Inorder
III. Postorder

(a) Any of the given traversal is sufficient
(b) Either I or III is sufficient
(c) I and III
(d) II and III

Q.12 If AVL tree has 15 nodes, what is the minimum and maximum possible height? Assume root is present at height 1?

(a) 4, 8 (b) 4, 5
(c) 3, 7 (d) 3, 8

Q.13 Consider the following code segment

```
struct node
{
    struct node *left;
    int data;
    struct node *right;
};
struct node *fun (struct node *P)
{
    If (P → right → right == NULL)
        P →  right = P → right → left;
    else
        P = fun (P → right);
    return (P);
}
```

What does the function fun do? Assume left subtree and right subtree of Binary search tree is not NULL.

(a) Finds the largest node in the binary search tree
(b) Finds the largest node in the binary search tree and deletes it
(c) Finds the smallest node in Binary search tree
(d) Finds the smallest node in binary search tree and deletes it

Q.14 A binary search tree was constructed by inserting following elements into an initially empty binary tree: 50, 27, 16, 88, 34, 65, 52, 77, 93, 4, 12, 29, 44, 92.

Preorder and postorder traversals of the resultant binary search tree were stored in arrays *A* and *B* respectively. How many elements have same index location in both the arrays? [Assume arrays *A* and *B* start from the same index]

Q.16 Let *T* be a binary search tree with 120 elements. What is the smallest possible height of *T*? Consider root is at height 0.

Q.17 Find the height of a tree for the following given traversals of the tree.

Inorder : *h, d, i, b, e, j, a, f, c, g, k*
Pre-order : *a, b, d, h, i, e, j, c, f, g, k*

Q.18 Let *T* be a *K*-ary tree (each internal node of *T* has at most *K* children). Suppose that the maximum depth of any node of *T* is d. What is the maximum number of leaves that *T* can have? [Assume root is at depth 0]

(a) K (b) Kd
(c) K^d (d) d^K

Q.19 If a tree has n_1 nodes of degree 1, n_2 nodes of degree 2 and n_3 nodes of degree 3, then which of the following holds?

(a) $n_1 = 2$ (b) $n_1 = n_3 + 2$
(c) $n_1 = n_3 - 1$ (d) $n_1 = n_2 + n_3$

Q.20 Consider the following recursive function.

```
bool f(Struct node *P)
{
    if(P = NULL) return TRUE;
    if(P → Left != NULL && Max(P → Left) > P →
    data)
        return FALSE;
    if(P → right != NULL && Min(P → right) <=
    P → data)
```

```
    return FALSE;
  if(! f(P → Left) || ! f(P → right))
    return FALSE;
  return TRUE;
}
```

Assume Max(q) function returns the maximum value from q and subtrees of q, Min(q) function returns the minimum value from q and subtrees of q. If root of the binary tree is passed to the function f( ), then what is the functionality of the above code?

(a) It checks if a given tree is a binary search tree or not.
(b) It checks if a given tree is a heap tree or not.
(c) Both (a) and (b)
(d) Neither (a) nor (b)

Q.21 A full 3-ary tree has 4 non-leaf nodes, how many leaf nodes does it have?

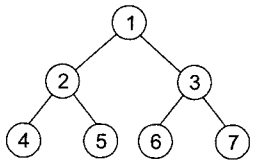
Q.22 Consider the following code.

```
find (*T1, *T2)
{   while(T1 ! = NULL && T2 ! = NULL)
    {   if ((T1 → data) = = (T2 → data))
        {   find (T1 → left, T2 → left);
            find (T1 → right, T2 → right);
        }
        else {printf ("GATE"); exit ( );}
    }
}
```

Find when "GATE" will not be printed by the above program? [Assume two binary trees are passed into the above function].

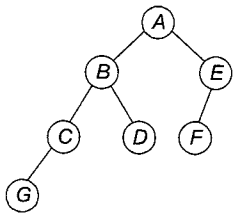
(a) If the two trees are same
(b) If the two trees are different
(c) If the two trees have same levels
(d) If the two trees have different levels

Q.23 Consider the following tree : If the post order traversal gives $ab - cd * +$ then the label of the nodes 1, 2, 3, ... will



(a) +, –, *, a, b, c, d
(b) a, –, b, +, c, *, d
(c) a, b, c, d, –, *, +
(d) –, a, b, +, *, c, d

Q.24 A balanced tree is given below



How many nodes will become unbalanced when a node is inserted as a child of node G?

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

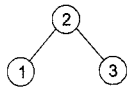
Q.25 In delete operation of binary search tree, we need inorder successor (or predecessor) of a node when a node to be deleted where it has both left and right child. Which of the following is true about inorder successor needed in delete operation?

(a) Inorder successor is always either leaf node or a node with empty right child.
(b) Inorder successor maybe an ancestor of the node.
(c) Inorder successor is always a leaf node.
(d) Inorder successor is always either a leaf node or a node with empty left child.

Q.26 Consider the following program.

```
void find (struct Node *node)
{   struct Node *ptr, *q;
    if (node == NULL) return;
    find (node → left);
    find (node → right);
    ptr = node → left;
    node → left = newNode (node → data);
    node → left → left = ptr;
}
```

If the root of following tree is passed to the above function, what is the level order traversal of output tree produced by above function? (newNode is a function which creates new node)



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

Q.27 The minimum size that an array may require to store a binary tree with 'n' nodes is _____.

(a) $2^{\lceil \log_2(n+1) \rceil} - 1$ (b) $2^n - 1$
(c) $2^n - n + 1$ (d) $n + 1$

Q.28 Consider the following Binary search tree with the following traversals.
Inorder: *A, M, N, O, P, Q, R, S, T, X, Z*
Preorder: *Q, P, A, M, N, O, X, S, R, T, Z*
Find the number of elements in the 4th, 5th and 6th level respectively. Assume root is at level 1.
(a) 3, 2, 2 (b) 2, 2, 2
(c) 3, 1, 1 (d) 3, 2, 1

Q.29 The depth of a complete binary tree with 'n' nodes is (log is to the base two)
(a) $\log(n+1) - 1$ (b) $\log(n)$
(c) $\log(n-1) + 1$ (d) $\log(n) + 1$

Q.30 The number of possible binary search trees with 3 nodes is
(a) 12 (b) 13
(c) 5 (d) 15

Q.31 A binary tree in which every non-leaf node has non-empty left and right subtrees is called a strictly binary tree. Such a tree with 10 leaf nodes
(a) cannot have more than 19 nodes
(b) has exactly 19 nodes
(c) has exactly 17 nodes
(d) cannot have more than 17 nodes

Q.32 Consider the following statements:
1. Infix, Prefix and Postfix notations for expressing sum of A and B are $A+B$, $+AB$, and $AB+$, respectively.
2. AVL tree is a binary tree in which the difference in heights between the left and the right sub tree is not more than one for every node.
3. Stack data structure is used to save and retrieve information in reverse order.
4. Queue data structure is known as LIFO.

Which of the statements given above are correct?
(a) 1, 2 and 3 (b) 2, 3 and 4
(c) 1, 3 and 4 (d) 1, 2 and 4

Q.33 A program takes an input a binary tree with 'n' leaf nodes and computes the value of function $g(x)$ for each node 'I'. If the cost of computing $G(x)$ = (Number of leaf nodes in left subtree of I-minimum number of node in right subtree of I), then the worst case time complexity of the program is
(a) $O(n)$ (b) $O(n^2)$
(c) $O(n \log n)$ (d) $O(\log n)$

Q.34 The key 14, 4, 6, 16, 32, 50 in the order are inserted into an initially empty AVL tree. Find total number of rotations to make AVL with the given keys. Assume "single rotation = 1 rotation" and "double rotation = 1 rotation".

Q.35 Let T be a binary search tree with 160 vertices. The smallest possible height of T is _____ (Assume root at height 0).

Q.36 Given pre-oder and post-oder traversal of binary search tree.
Pre-oder : 60, 37, 25, 21, 23, 39, 38, 50, 70, 65, 100, 85, 110.
Post-oder : 23, 21, 25, 38, 50, 39, 37, 65, 85, 110, 100, 70, 60
The maximum number of node presents at any level is _____.

■■■■

Answers Trees

1.	(d)	2.	(c)	4.	(d)	5.	(b)	6.	(c)	7.	(c)	8.	(d)	9.	(b)	10.	(b)
11.	(b)	12.	(b)	13.	(b)	18.	(c)	19.	(b)	20.	(a)	22.	(a)	23.	(a)	24.	(b)
25.	(d)	26.	(b)	27.	(a)	28.	(c)	29.	(a)	30.	(c)	31.	(b)	32.	(a)	33.	(a)

Explanations Trees

1. (d)
In this particular case it is BST and search time is proportional to height (H).

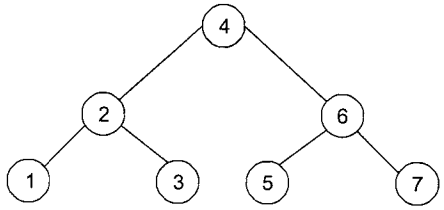
2. (c)
By the method of elimination : Full binary tree looses its nature when operation of insertion and deletion are done. So complete binary tree is better one because for incomplete binary tree, extra storage is required and overhead of NULL node checking takes place.

3. (3)
The In-order traversal is
HFIEJGZ and postorder is
HIFJZGE so left subtree
Contain 3 nodes

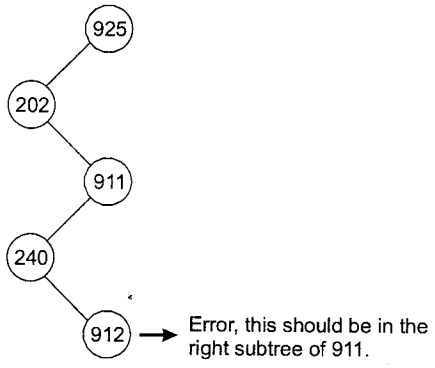
4. (d)
Sum of degree of all the nodes will be
$= 1 * n_1 + 2 * n_2 \ldots m * n_m$ ...(1)
We know sum of degree of all the nodes is equal to number of edges ...(2)
Number of edges = Number of nodes – 1 ...(3)
Total number of nodes = $n_0+n_1 + n_2+\ldots n_m$ ...(4)
from (1), (2), (3), (4) we have
$1*n_1+2*n_2\ldots m * n_m = (n_0 + n_1 + n_2 +\ldots n_m) - 1$
$\Rightarrow n_0 = 1+[1*n_2(2-1) + n_3(3-1)\ldots n_m(m-1)]$

5. (b)
At leaf-level.
1. Group the nodes into 3 sets of 3, it gives 4 internal nodes.
2. Group the nodes into 3 sets of 2 and 1 set of 3, it gives 7 internal nodes.

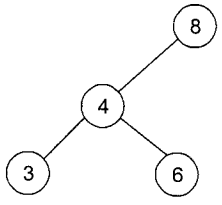
6. (c)



7. (c)

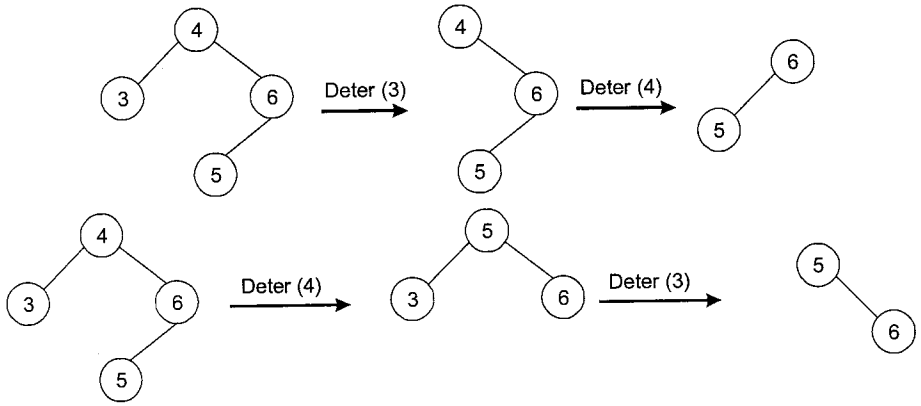


8. (d)
Counter example of (1)

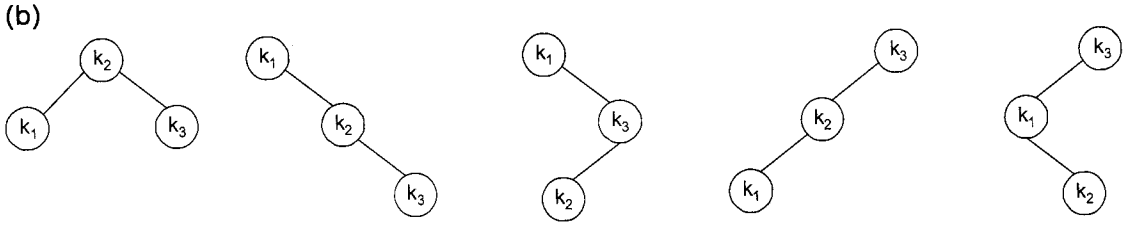


Set B = {8, 4, 3}
Set C = {6}
but 6 > 4 and 6 > 3

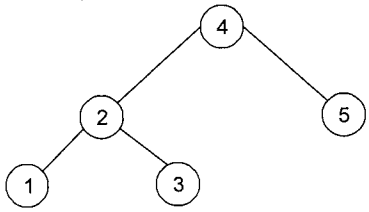
Counter example of (2)



9. (b)



10. (b)

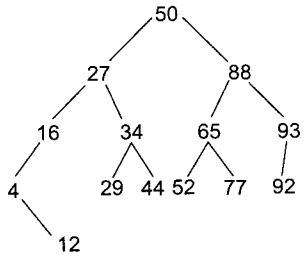


11. (b)
To construct BST either preorder or postorder is sufficient because inorder is always sorted for BST.

12. (b)
$\lfloor \log_2(n) \rfloor$ Minimum height = 4
$\lceil \log_2(n) \rceil$ Maximum height = 5

13. (b)
The given code will deletes the largest element of the binary search tree, since the largest element is always found at right.

14. (3)



Array A: Preorder:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	50	27	16	4	12	34	29	44	88	65	52	77	93	92

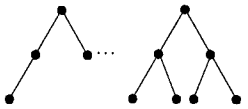
Array B: Postorder:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	12	4	16	29	44	34	27	52	77	65	92	93	88	50

The elements **16, 34, 65** have same location in A and B.

16. **(6)**

Complete binary tree can make smallest possible height of Binary Search Tree (T).

$\Rightarrow \quad 2 \le n \le 3 \Rightarrow h = 1$ (smallest)



$\Rightarrow \quad 4 \le n \le 7 \Rightarrow h = 2$

Similarly $8 \le n \le 15 \Rightarrow h = 3$

$\therefore$ Smallest height possible: [when complete tree]

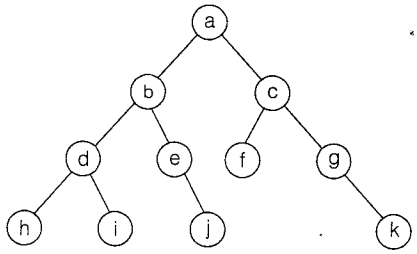
$$h = \left\lceil \log_2^{(n+1)} \right\rceil - 1$$

$\therefore$ Largest height possible: [When skew tree]

$h = n - 1$

Smallest possible height $= \left\lceil \log_2^{(120+1)} \right\rceil - 1 = 7 - 1 = 6$

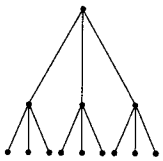
17. **(3)**



Height = 3

18. **(c)**

Example: 3-ary tree



$d = 0 \Rightarrow 3^0 = 1$ node (root)
$d = 1 \Rightarrow 3^1 = 3$ leaf nodes
$d = 2 \Rightarrow 3^2 = 9$ leaf nodes, etc.

At depth $d \Rightarrow$ The maximum number of nodes $= 3^d$.
For K-ary tree $\Rightarrow$ The maximum number of nodes at depth $d = K^d$.
If d is maximum depth of any node of T, then maximum number of leaves $= K^d$.

19. (b)

Total nodes (n) = $n_1 + n_2 + n_3$

Total edges (e) = $(n_1+n_2+n_3) - 1$ [$\because e = n-1$]

Sum of degrees of vertices = $2e$

$\Rightarrow 1 * n_1 + 2 * n_2 + 3 * n_3 = 2e$

$\Rightarrow n_1 + 2n_2 + 3n_3 = 2e$ [$\because e = n_1 + n_2 + n_3 - 1$]

$\Rightarrow n_1 + 2n_2 + 3n_3 = 2(n_1 + n_2 + n_3 - 1)$

$\Rightarrow n_1 + 2n_2 + 3n_3 = 2n_1 + 2n_2 + 2n_3 - 2$

$\Rightarrow n_1 = n_3 + 2$

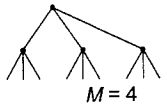
20. (a)

Given function $f( )$ checks whether a given tree is BST or not. It returns 'TRUE' if a tree is BST, otherwise it returns 'FALSE'.

21. (9)

leaf nodes = $N * (M - 1) + 1$

$= 4 * (3 - 1) + 1 = 9$



22. (a)

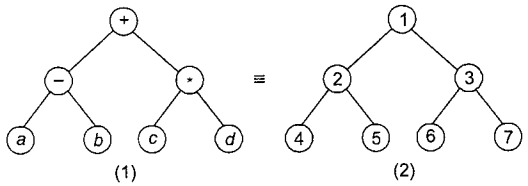
In the given code two trees goes left simultaneously and it can go right also. Those two trees if they have same data at every node then it checks for left and right. If incase any mismatch of data then the program will exit by printing GATE

So, the function will not print if the given two trees are same.

Hence option (a) is correct.

23. (a)

Post order = $ab - cd * +$ according to given postorder the binary tree is



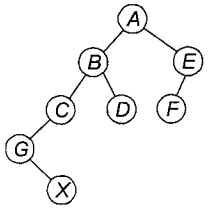
(1) (2)

According to tree (1) and (2)

1, 2, 3, 4, 5, 6, 7 ≡ +, −, *, a, b, c,d.

24. (b)

Let a node (x) is inserted

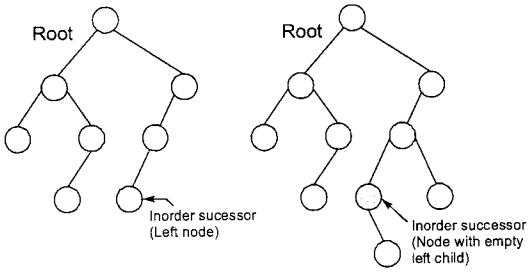


Node A, B, C height = 2

They are unbalanced.

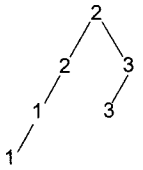
25. (d)

Successor of root element is always the smallest element of the Right subtree. Because it will be the next largest element after the elemetn to be deleted.



26. (b)

Tree after execution of above code



Level order traversal is 2 2 3 1 3 1.

27. (a)

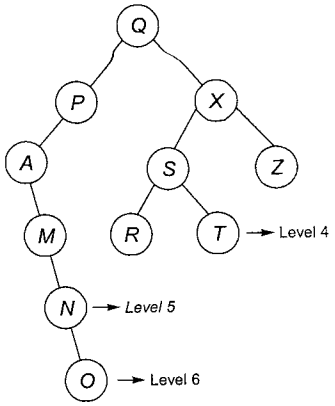
In case of full or complete binary tree minimum height $\Rightarrow h_{min} = \lceil \log_2(n+1) \rceil$

Hence, last element will be stored at $2^{h_{min}} - 1$

$\therefore$ Minimum size = $2^{\lceil \log_2(n+1) \rceil} - 1$

28. (c)

The resultant BST obtained is



$\therefore$ The number of elements in the 4th, 5th and 6th level are. 3, 1, 1

29. (a)

If the depth is d, the number of nodes n will be $2^{(d+1)} - 1$.

So, $n + 1 = 2^{(d+1)}$ or $d = \log(n + 1) - 1$

30. (c)

Number of possible binary search trees with n nodes $= \dfrac{2n_{C_n}}{n+1}$

$\therefore$ For 3 nodes $= \dfrac{2 \times 3_{C_3}}{3+1} = \dfrac{6_{C_3}}{4} = 5$

31. (b)

A strictly binary tree with 'n' leaves must have $(2n - 1)$ nodes. Verify for some small 'n'. This can be proved by the principle of mathematical induction.

32. (a)

Queue data structure is known as FIFO (First In First Out).

While

Stack is known as LIFO (Last In First Out).

Statements 1, 2 and 3 are correct.

33. (a)

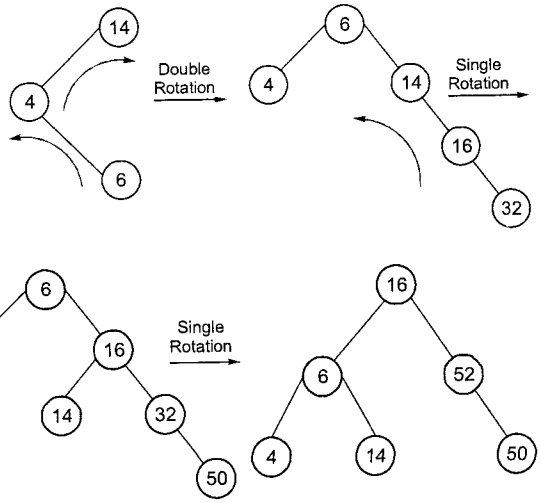
$G(x)$ = (Number of leaf nodes in left – Number of leaf nodes in right subtree of I).

To find number of leaf nodes in left and right subtree of node I will take $O(n)$ time. One time visit of each node in left subtree and one time visit of each node in right.

Difference will take constant time.

So, overall time complexity = $O(n)$.

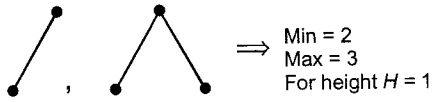
34. (3)

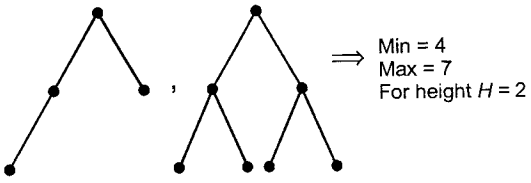


One double and two single rotations are required.

So total 3 rotations.

35. (7)





Similarly for height $H = 3$

Minimum = 7

Maximum = 15

$\therefore$ Smallest height possible (when complete tree)

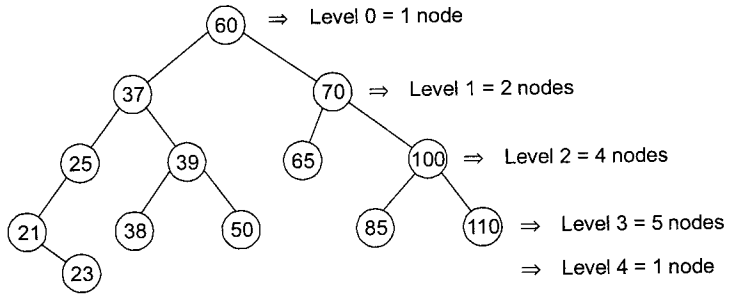
$$H = \lceil \log_2(n + 1) \rceil - 1$$

So height $H = \lceil \log_2(160 + 1) \rceil - 1$

$= 8 - 1 = 7$

36. (5)

In-oder of binary search tree is in sort from so, in-oder = 21,23,25,37,38,39,50,60,65,70,85,100,110.
Now by using in-oder and pre-oder, the BST constructed is:



So, maximum nodes are 5 which is at level 3.

■■■■

7 CHAPTER

Hashing Techniques

Q.1 A hash function *f* defined as *f*(key) = key mod 7, with linear probing, insert the keys 37, 38, 72, 48,98, 11, 56.
Data 11 and 56 will be stored on at the locations _____ , _____ respectively (hash address starts from '0' and location start from '1')
(a) 6, 1 (b) 5, 1
(c) 5, 2 (d) 6, 2

Q.2 A hash function f defined as *f*(key) = key mod 7, with linear probing, is used to insert the keys 37, 38, 72, 48, 98, 11, 56, into a table indexed from 0 to 6. What will be the location of key 11?
(a) 3 (b) 4
(c) 5 (d) 6

Q.3 A hash table with 10 buckets with one slot per bucket is depicted in Figure the symbols, *S*1 and *S*7 are initially entered using a hashing function with linear probing.
The maximum number of comparisons needed in searching an item that is not present is

0	*S*7
1	*S*1
2	
3	*S*4
4	*S*2
5	
6	*S*5
7	
8	*S*6
9	*S*3

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

Q.4 A hash table can store a maximum of 10 records. Currently there are records in locations 1, 3, 4, 7, 8, 9, 10. The probability of a new record going into location 2, with a hash function resolving collisions by linear probing is
(a) 0.6 (b) 0.1
(c) 0.2 (d) 0.5

Q.5 Consider a hashing function that resolves collision by quadratic probing. Assume the address space is indexed from 1 to 8. Which of the following locations will never be probed if a collision occurs at position 4?
(a) 4 (b) 5
(c) 8 (d) 2

Q.6 A hash table has space for 100 records. What is the probability of collision before the table is 10% full?
(a) 0.45 (b) 0.5
(c) 0.3 (d) 0.34 (approximately)

Q.7 Given keys (x): 3, 23, 1, 0, 15, 31, 4, 7, 11, 43
M (Hash table size) = 11(location 0 to location 10) and initially hash table is empty.
Find number of collisions occured in hash table if x mod M hash function is used to map the given keys.

Q.8 Assume the given hash table uses linear probing and hash function $h(x) = x$ mod 9.
If keys 22, 11, 17, 13, 16, 12, 20 are inserted into the hash table in the order, find the total number of collisions occurred in the table. (hash table locations are 0, 1, 2, ...,8)

Q.9 Suppose we used a hash function *H*(*n*) to hash '*n*' distinct elements (keys) into an array *T* of length '*m*'. What is the expected number of colliding pairs of elements, if we used simple uniform hashing?

(a) $\Theta(n^2)$ (b) $\Theta(m^2)$
(c) $\Theta(n^2/m)$ (d) $\Theta(n^3/m^2)$

Q.10 Given the input sequence {11, 33, 43, 99, 34, 79, 19} and hash table of size 10 with the hash function $h(k) = k \bmod 10$. If hash table uses quadratic probing, the number of collisions occurred while mapping the given sequence is_____.

Q.11 Consider the hash table of size 12 that uses open addressing with linear probing. Let $h(k) = k \bmod 12$ be the hash function used. A sequence of records with keys 43, 63, 84, 11, 5, 72, 15, 16 is stored into an initially empty hash table, the bins of which are indexed from zero to 12. The index of the bin into which last element inserted is_____.

Q.12 Consider the following keys that are hashed into the hash table in the order given using the hash function $H(i) = (3i + 5) \bmod 11$.

12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5

Where to handle the collision, chaining is used, after inserting all the above keys in table if new key is inserted into table then what is the probability new item is hashed into empty slot?

(a) 0.36 (b) 0.40
(c) 0.50 (d) 0.76

■■■■

Answers Hashing Techniques

1. (d) 2. (c) 3. (b) 4. (a) 5. (d) 6. (a) 9. (c) 12. (a)

Explanations Hashing Techniques

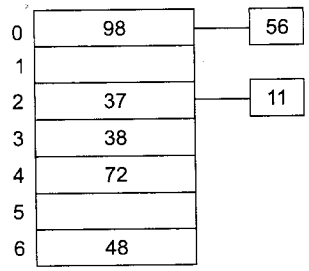
1. (d)

f(key) = key mod 7

	Remainder	Location
$f(37) = 37 \bmod 7$	2	3
$f(38) = 38 \bmod 7$	3	4
$f(72) = 72 \bmod 7$	2	(location 2 is not vacant move to next vacant location) (5)
$f(48) = 48 \bmod 7$	6	7
$f(98) = 98 \bmod 7$	0	1
$f(11) = 11 \bmod 7$	4	5
$f(56) = 56 \bmod 7$	0	2

Remainder	0	1	2	3	4	5	6
	98	56	37	38	72	11	48
Locations	1	2	3	4	5	6	7

2. (c)



3. (b)

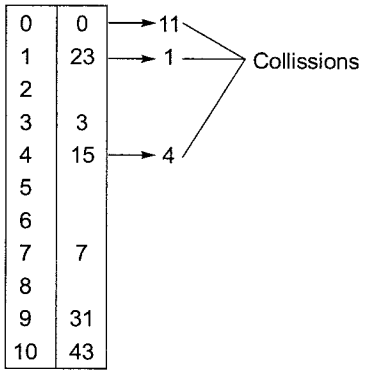
It will be one more than the size of the biggest cluster (which is 4) in this case. This is because, assume a search key hashing onto bin 8. By linear probing the next location for searching is bin 9. Then 0, then 1. If all these resulted in a miss, we try at bin 2 and stop as it is vacant. This logic may not work if deletion operation is done before the search.

4. (a)
If the new record hashes onto one of the six locations 7, 8, 9, 10, 1 or 2, the location 2 will receive the new record. The probability is 6/10 (as 10 is the total possible number of locations).

5. (d)
You can verify that the 1st, 3rd, 5th, 7th...probes check at location 5.
The 2nd, 6th, 10th...probes check at location 8.
The 4th, 8th, 12th...probes check at location 4.
The rest of the address space will never be probed.

6. (a)
If there is only one record, then the probability of a collision will be 1/100. If 2, then 2/100 etc., If 9 then 9/100. So, the required probability is
$1 + 2 + 3 \ldots 9/100 = 0.45$.

7. (3)



8. (5)

		11	12	22	13	20	16	17
0	1	2	3	4	5	6	7	8

Key 13 has 1 collision
Key 20 has 4 collisions
$\therefore$ Total 5 collisions.

9. (c)
Let $X_{i,j}$ be an indicator random variable equal to 1 if element i and j collide and equal to 0 otherwise. Simple uniform hashing is used i.e., the probability of element i-hashing to slots K is 1/m. Therefore the probability that i and j both hashed to same slot $Pr(X_{i,j}) = 1/m$.
Hence expected $[X_{i,j}] = 1/m$.

The E [no. of collision pairs] $= E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{i,j}\right]$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{i,j}]$$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n} 1/m$$

$$= \frac{n(n+1)}{2m} = \Theta(n^2/m)$$

10. (6)

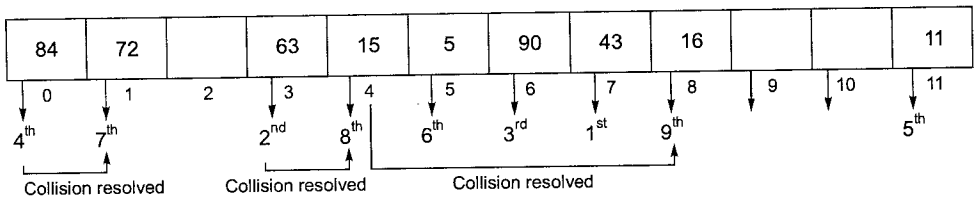
79	11		33	43	34			19	99
0	1	2	3	4	5	6	7	8	9

$h(43) = 3$ $\Rightarrow$ Collision
$= 3 + 1^2 = 4$
$h(34) = 4$ $\Rightarrow$ Collision
$= 4 + 1^2 = 5$
$h(79) = 9$ $\Rightarrow$ Collision
$= 9 + 1^2 = 0$
$h(19) = 9$ $\Rightarrow$ Collision1
$= 9 + 1^2 = 0$ $\Rightarrow$ Collision2
$= 9 + 2^2 = 3$ $\Rightarrow$ Collision3
$= 9 + 3^2 = 8$

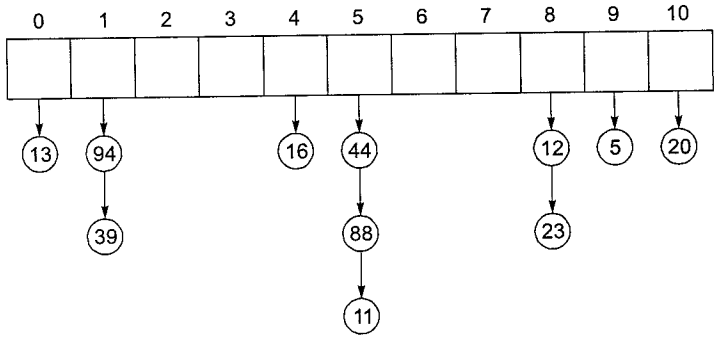
11. (8)

Sequence: 43, 63, 84, 11, 5, 72, 15, 16

Hash function $\Rightarrow$ $h(k) = k \bmod 12$



12. (a)



Empty slots = 4

So, probability of next item to be inserted into 4 free slots out of 11 slots is

$$\frac{4}{11} = 0.363 \cong 0.36$$

■■■■