# POSTAL BOOK PACKAGE 2025

# CONTENTS

## COMPUTER SCIENCE & IT

**Objective Practice Sets**

## Programming and Data Structures

# 1 CHAPTER
# Programming Methodology

## Multiple Choice Questions & NAT Questions

1. Consider the following function declaration

int* *f*(int *);

Which of the following is correct about the declaration?

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

2. Find the output of the following program:

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

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

3. Consider the following code?

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

How many zero's are printed in the output?

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

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

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

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

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

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

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

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

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

**7.** Consider the following code:

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

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

**8.** What is output of the following program?

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

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

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

**List-I**

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

**List-II**

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

**Codes:**

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

**10.** Consider the following pseudocode program:

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

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

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

**11.** Consider the following code:

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

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

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

## Answers Programming Methodology

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

## Explanations Programming Methodology

**1. (b)**

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

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

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

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

**2. (a)**

**Linker error:** Undefined symbol-*i*

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

**3. (4)**

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

So total 4 times zero is printed.

**4. (d)**

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

∴ 65 + 1 = 66 (B)

65 + 2 = 67(C)

65 + 3 = 68(D)

∴ *BCD* is the correct output.

**5. (a)**

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

∴ $i << 3 \Rightarrow i * 8$

$j >> 2 \Rightarrow j / 4$

**6. (a)**

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

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

$n >> = 1$

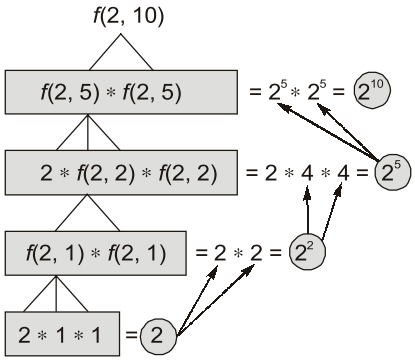
}

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

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

**7. (1024)**

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




**8. (d)**

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

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

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

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

**9. (b)**

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

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

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

**10. (d)**

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

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

**11. (c)**

$X$ : $b + c * 2$ is 8

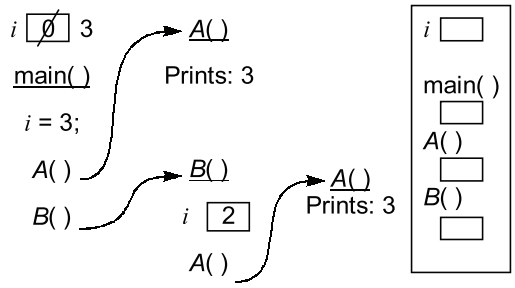
Case 8: prints 3 then default case prints 2

∴ Output prints 32.

**12. (c)**

Return statement can contain an expression.

**13. (d)**



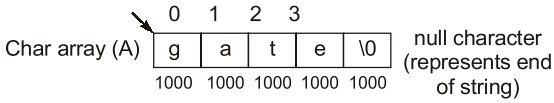

Output printed by the code: 3, 3

**14. (c)**

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

**15. (a)**

Let string gate be stored from memory location 1000.




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

**16. (d)**

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

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

**17. (b)**

After Execution

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

$a = 20$, $b = 30$

# 2 CHAPTER
# Arrays

## Multiple Choice Questions & NAT Questions

**1.** Which of the following C expressions access the $(i, j)^{th}$ entry of an $(m \times n)$ 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) $n \times (m - i) + j$

**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].

**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

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

**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

**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).

**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

**8.** Let's look about the algorithm:

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

for ($j = i - 1$; $j \geq 0$ && ($A([j] > \text{temp}$); $j--$)
$A[j + 1] = A[j]$;
$A[j] = \text{temp}$;
}

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) $O(\log_2 n)$

**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

**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) $O(n^2)$

**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) $O(n)$

**12.** What is the output of the following C code? Assume that the address of $X$ is 2000 (in decimal) and an integer requires four bytes of memory.

```
int main( ).
{
unsigned int x [4] [3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9},
{10, 11, 12}}
printf("%u,%u,%u", X + 3, * (X + 2) + 3);
}
```

(a) 2036, 2036, 2036
(b) 2012, 4, 2204
(c) 2036, 10, 10
(d) 2012, 4, 6

**13.** Consider the following C function:

```
#include <stdio.h>
int main(void)
{
    char c[ ] = "ICRBCSIT17";
    char *p = c;
    printf("%s", c + 2[p] – 6[p] – 1);
    return 0;
}
```

The output of the program is

(a) SI (b) IT
(c) T1 (d) 17

**14.** The output of the following program is

```
main ( )
{   static int x[ ] = {1, 2, 3, 4, 5, 6, 7, 8};
    int i ;
    for (i = 2; i < 6; ++i)
    x[x[i]] = x[i];
    for (i = 0; i < 8; ++i)
    printf("%d", x[i]);
}
```

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

**15.** Which of the following is true?

(a) In sorted array of '$n$' distinct elements, deletion of an element from beginning takes $O(\log n)$ time.
(b) In sorted array of '$n$' distinct elements, insertion of an element takes $O(\log n)$ time.
(c) In sorted array of '$n$' distinct elements, finding $i^{th}$ largest element take $O(1)$ time.
(d) In unsorted array of '$n$' distinct elements, insertion of an element take $\Omega(\log n)$ time.

**16.** Consider the function given below, which should return the index of first zero in input array of length '$n$' if present else return –1.

```
int index of zero (int[ ] array, int n) {
for (int i = 0; [P] ; i++);
    if (i = = n)
        return –1;
    return i;
}
```

Which of the should be place in code at [P], so that code will work fine?

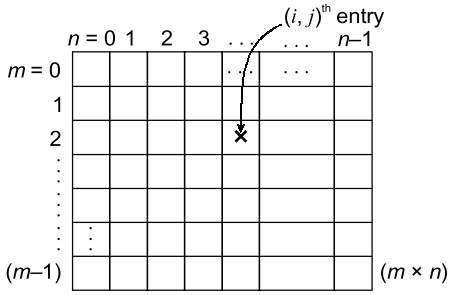
(a) array[$i$]! = 0 && $i \leq$ n
(b) array[$i$]! = 0 && $i <$ n
(c) ! array[$i$] = 0 && $i <$ n
(d) ! array[$i$] = = 0|| $i <$ n

## Answers Arrays

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1.** | (b) | **2.** | (23699) | **3.** | (b) | **4.** | (c) | **5.** | (a) | **6.** | (63) | **7.** | (d) | **8.** | (c) | | |
| **9.** | (b) | **10.** | (b) | **11.** | (a) | **12.** | (a) | **13.** | (d) | **14.** | (a) | **15.** | (c) | **16.** | (b) | **17.** | (b) |
| **18.** | (3750) | | | **19.** | (c) | **20.** | (b, c) | | | | | | | | | | |

## 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)**

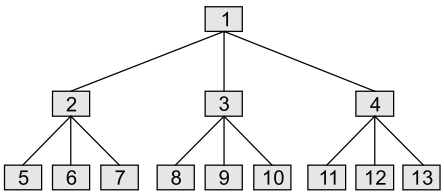
| | $i$ | $j$ | $k$ |
|---|---|---|---|
| Let | $A[r_1]$ | $[r_2]$ | $[r_3]$ |
| | 90 | 30 | 40 |
| | (planes) | (rows) | (colums) |

**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)**

| $(\ell)$ | | | | $(m)$ | | | | $(h)$ |
|---|---|---|---|---|---|---|---|---|
| 12 | 14 | 15 | 15 | 15 | 18 | 110 | 120 | 125 |

| $(\ell)$ | | $(m)$ | | $(h)$ |
|---|---|---|---|---|
| 12 | 14 | 15 | 15 | 15 |

| $(\ell)$ | $(m)$ | $(h)$ |
|---|---|---|
| 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.

[$l_{bj}$ → 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 → $n \log n$.
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++;
}
```

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

**12. (a)**

Array is stored as follows:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1<br>2000 | 2<br>2004 | 3<br>2008 |
| 1 | 4<br>2012 | 5<br>2016 | 6<br>2020 |
| 2 | 7<br>2024 | 8<br>2028 | 9<br>2032 |
| 3 | 10<br>2036 | 11<br>2040 | 12<br>2044 |

$x + 3$ → Skips 3 rows, i.e. 9 elements

$2000 + 9 * 4 = 2036$

$*(x + 3)$ → selects 3rd row ⇒ 2036

$*(x + 2) + 3$ → selects 2nd row and skips 3 elements ⇒ $2024 + 3 * 4 = 2036$.

**13. (d)**

C stores the base address of string "ICRBCSIT17"

| I | C | R | B | C | S | I | T | 1 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 | 1008 | 1009 |

2[$P$] 6[$P$]

$C$ 1000

$P$ 1000

$= 1000 + 'R' - 'I' - 1$

$= 1000 + 9 - 1 = 1008$

It prints string starting from 1008, i.e. 17.

**14. (a)**

$x[\,] =$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

$i = 2,\ x[x[2]] = x[2] \Rightarrow x[3] = 3$

$i = 3,\ x[x[3]] = x[3] \Rightarrow x[3] = 3$

$i = 4,\ x[x[4]] = x[4] \Rightarrow x[5] = 5$

$i = 5,\ x[x[5]] = x[5] \Rightarrow x[5] = 5$

$x[\,] =$

| 1 | 2 | 3 | 3 | 5 | 5 | 7 | 8 |
|---|---|---|---|---|---|---|---|

**15. (c)**

- In sorted array, deletion from beginning takes $O(n)$ time as after deletion operation, rest of the elements have to be shifted.
- In sorted array, insertion of an element takes $O(n)$ time as the element's correct position has to be reached and rest elements have to be shifted.
- In sorted array, finding $i^{th}$ largest or smallest element take $O(1)$ time as elements are arranged in ascending or descending order.
- In unsorted array, insertion takes $O(1)$ time as element can be inserted at last.

# 3 CHAPTER

# Stack

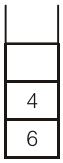
## Multiple Choice Questions & NAT Questions

1. Computes the postfix equivalent of following expression:

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

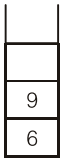
(a) 3$x$1 + log∗$a$2/- (b) 31$x$ + log ∗ $a$2/-
(c) 31$x$ log + ∗$a$2/- (d) 31 log$x$ + ∗ $a$22/-

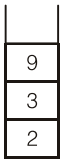
2. In evaluating the arithmetic expression 2 ∗ 3 – (4 + 5), using stack to evaluate.
Which of the following is stack configuration is not possible

(a)



(b)



(c)



(d)



3. 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( ));

4. 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

5. 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.

6. 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) $ab + pq + rst * * \uparrow\uparrow$

7. 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

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

9. 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?

10. 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) $AB \$ C - D * EF / GH / ++$

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

(a) $- / * AB * (C - D)$

(b) $- ABCD * DE \$$

(c) $- A / B * C \$ DE$

(d) $- A / BC * \$ DE$

12. 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 +

13. 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

## Answers Stack

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1.** | (a) | **2.** | (d) | **3.** | (b) | **4.** | (c) | **5.** | (2) | **6.** | (c) | **7.** | (10) | **8.** | (d) | **9.** | (1) |
| **10.** | (a) | **11.** | (c) | **12.** | (8) | **13.** | (23) | **14.** | (15) | **15.** | (d) | **16.** | (a) | **17.** | (c) | **18.** | (b) |
| **19.** | (14) | **20.** | (1) | **21.** | (4) | **22.** | (d) | **23.** | (10100100) | **24.** | (a) | **25.** | (b) | **26.** | (b) | | |
| **27.** | (11) | **28.** | (a) | **29.** | (a) | **30.** | (a, d) | **31.** | (a, b, c) | **32.** | (b, c, d) | **33.** | (a, b, d) | | | | |
| **34.** | (a, c) | **35.** | (a, c, d) | **36.** | (a, c) | **37.** | (a, c) | **38.** | (a, c) | **39.** | (a, c, d) | | | | | | |

## Explanations Stack

**1. (a)**

| Input | Postfix Expression | Stack |
|---|---|---|
| 3 | 3 | |
| * | 3 | * |
| log | 3 | *log |
| ( | 3 | *log( |
| $x$ | 3$x$ | *log( |
| + | 3$x$ | *log(+ |
| 1 | 3$x$1 | *log(+ |
| ) | 3$x$1+ | *log |
| – | 3$x$+log* | – |
| $a$ | 3$x$1+log* $a$ | – |
| / | 3$x$1+log* $a$ | –/ |
| 2 | 3$x$1+log* $a$2 | –/ |

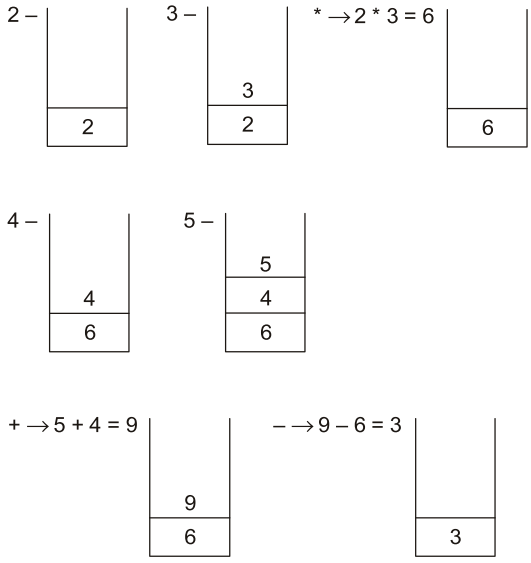
Final output → 3$x$+ log* a2/ –

**2. (d)**

First converting the infix expression to postfix.

| Input | Stack | 'Postfix' |
|---|---|---|
| 2 | | 2 |
| * | * | 2 |
| 3 | * | 23 |
| – | – | 23 * |
| ( | –( | 23 * |
| 4 | –( | 23 * 4 |
| + | –(+ | 23 * 4 |
| 5 | –(+ | 23 * 45 |
| ) | – | 23 * 45 + |
| Postfix → | 23 * 45 + – | |

Evaluating the postfix expression–




Hence (d) is not possible stack configuration.

**3. (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( );

**4. (c)**

Given postfix expression is evaluated using the stack.

**5. (2)**

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

∴ Substring: /+ has only operators and two distinct operators.