CLASS TEST

Sl. : 01JPCS-PDS-19072023

MADE EASY
India's Best Institute for IES, GATE & PSUs

**Delhi | Bhopal | Hyderabad | Jaipur | Pune | Bhubaneswar | Kolkata**

**Web:** www.madeeasy.in | **E-mail:** info@madeeasy.in | **Ph:** 011-45124612

# COMPUTER SCIENCE & IT

## Programming and Data Structures

**Duration : 1:00 hr.** **Maximum Marks : 50**

Read the following instructions carefully

1. This question paper contains **30** objective questions. **Q.1-10** carry one mark each and **Q.11-30** carry two marks each.
2. Answer all the questions.
3. Questions must be answered on Objective Response Sheet (**ORS**) by darkening the appropriate bubble (marked **A**, **B**, **C**, **D**) using HB pencil against the question number. Each question has only one correct answer. In case you wish to change an answer, erase the old answer completely using a good soft eraser.
4. There will be **NEGATIVE** marking. For each wrong answer **1/3rd** of the full marks of the question will be deducted. More than one answer marked against a question will be deemed as an incorrect response and will be negatively marked.
5. Write your name & Roll No. at the specified locations on the right half of the **ORS**.
6. No charts or tables will be provided in the examination hall.
7. Choose the **Closest** numerical answer among the choices given.
8. If a candidate gives more than one answer, it will be treated as a **wrong answer** even if one of the given answers happens to be correct and there will be same penalty as above to that questions.
9. If a question is left blank, i.e., no answer is given by the candidate, there will be **no penalty** for that question.

**DO NOT OPEN THIS TEST BOOKLET UNTIL YOU ARE ASKED TO DO SO**

**Q.No. 1 to Q.No. 10 carry 1 mark each**

**Q.1** 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.2** Which of the following permutations can be obtained in the output (in the same order) using a stack assuming that the input is the sequence 5, 7, 8, 4, 6 in that order?

(a) 6, 8, 4, 7, 5 (b) 6, 4, 5, 7, 8
(c) 6, 4, 7, 8, 5 (d) 7, 8, 4, 6, 5

**Q.3** Consider the following program.

```
main ( )
{
    int i, j;
    int A[m] [n] = {{1, 2, 3} {4, 5, 6}}
    for (i = 0; i < n; i ++)
    for (j = 0; j < m; j ++)
    print f("%d", * (A[j] + i));
}
```

For the output printed by the above program.

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

**Q.4** Which of the following statements is correct:

(a) Void pointers can be used for dereferencing.
(b) Arithmetic operations can be applied on void pointers.
(c) The default value of extern storage class is garbage value.
(d) A particular extern variable can be declared many times, but canbe initialized at only 1 time.

**Q.5** Consider '*T*' to be a '*n*' - ary tree (each internal node has at most '*n*' children). Suppose the depth of '*T*' is '*k*'. Which of the following correctly represents the maximum number of leaves that '*T*' can have ?

(a) $n^k$ (b) $k^n$
(c) $nk$ (d) None of these

**Q.6** An $n \times n$ where '*n*' ranging from {1 to *n*} matrix '*r*' is defined as follows :

$r[i, j] = i$, if $(i = = j)$
$r[i, j] = i^2 - j^2$, if $(i < j)$
$r[i, j] = j^2 - i^2$, if $(i > j)$

The sum of the elements of the array '*r*' is

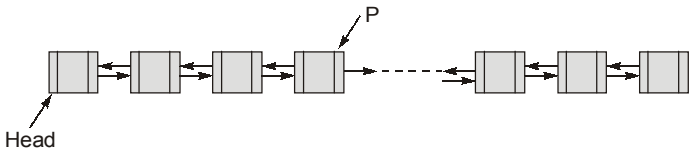
(a) 0 (b) $n^2$
(c) $\frac{n(n+1)}{2}$ (d) None of these

**Q.7** Let '*T*' be a rooted Binary Tree above vertices are labelled with symbols 1, 2, 3, 4, 5, 6, 7, 8. Suppose the inorder and postorder traversal of *T* produces the following sequences.

In-order : 1, 2, 3, 4, 5, 6, 7, 8
Post-order : 1, 3, 2, 5, 6, 8, 7, 4
Which of the following is correct with respect to *T* ?

(a) '*T*' has 5 leaf nodes
(b) Right subtree of the root node satisfies AVL property
(c) '*T*' satisfies basic heap properties
(d) None of these

**Q.8** Consider a containing *n* nodes given below :




We want to insert an new node in the given linked list after node P. Which of the following sequence of operation is correct ?

(a) 1. new → next = P → next;
2. new = P → next;
3. new → prev = P;
4. (P → next) → prev = next;

(b) 1. new → next = P → next;
2. new → next = P;
3. new → prev → P;
4. (new → next) → prev = new;

(c) 1. new → next = P → next;
2. P → next = new;
3. new → prev = P;
4. (new → next) → prev = new;

(d) 1. new → next → next;
2. P → next = new;
3. new → prev = P;
4. (P → next) → prev = new;

**Q.9** A stack is implemented using a circular doubly linked list such that PUSH and POP operations are performed efficiently. Which one of the following statements can be correct ?
(a) Both operations can be performed in O(1) time.
(b) The worst case time complexity for both operations will be $\Omega(n)$.
(c) Stack can not be implemented with gives data structure.
(d) None of these

**Q.10** Consider the following C program

```
main ( )
  {
      int p = –3, q = 2, r = 0, s, t ;
      s = ++ p && ++ q || r ++ ;
      t = p + q + s ++ ;
      printf ("\ n % d % d", s, t) ;
  }
```

Which of the following represents output of above program ?
(a) 1 2
(b) 0 2
(c) 1 3
(d) 0 3

**Q.No. 11 to Q.No. 30 carry 2 marks each**

**Q.11** Consider the following C program segment:

```
# include <stdio.h>
main( )
{
    static char *s[ ] = {"madeeasy", "online", "test", "series"};
    char **ptr[ ] = {s + 3, s + 2, s + 1, s}, *** p;
    p = ptr;
```

```
    ++p;
    Printf("%s", * – – *++p + 3)
}
```

What will be printed by the program?

(a) line (b) ies
(c) test (d) eeasy

**Q.12** Consider the following code on binary tree where p and q are two given nodes of that tree.

```
Struct node
{
    int data;
    struct node * left;
    struct node * right;
}
typedef node NODE;
NODE find (NODE root, NODE p, NODE q)
{
    if (!root)
    return NULL;
    if (root == p || root == q)
    return root;
    else
    {
        NODE l = find (root → left, p, q);
        NODE r = find (root → right, p, q);
        if (l && r)
            return root;
        else if (l)
            return l;
        else if (r)
            return r;
        else if (l = r = NULL)
            return NULL;
    }
}
```

The value returned by the function "find" when a pointer to the root of a non-empty tree is passed as argument is

(a) The lowest common ancestor of two node.
(b) The highest common ancestor of two nodes.
(c) The largest common successor of two nodes.
(d) Node of these

**Q.13** Consider the AVL tree 'T' in which left subtree contain quarter of the maximum number of nodes possible in the balanced AVL tree of height 'h' and right subtree consist of one fifth of the maximum number of nodes possible in AVL tree of height 'h'.

Assume that tree 'T' may or may not be height balanced at present. What is the total maximum possible number of nodes in 'T'.

(a) $\frac{7}{20}\left(2^{h+1}-1\right)-1$ (b) $\frac{9}{20}\left(2^{h+1}-1\right)+1$

(c) $\frac{7}{30}\left(2^{h+1}+1\right)$ (d) $\frac{9}{30}\left(2^{h+1}+1\right)-1$

**Q.14** Consider the following function that comutes the value of $\binom{m}{n}$ correctly for all legal values m and n (m ≥ 1, n ≥ 0) 0 and m > n)

```
int func (int m, int n)
{
    if [(n == 0) || (m == n) return 1;
    else return (E);
}
```

In the function, which of the following is the correct expression for E?

(a) func (m – 1, n) + func (m – 1, n – 1) (b) func (m – 1, n + 1) + func (m – 1, n)

(c) func (m, n) + func (m, n – 1) (d) None of these

**Q.15** What will be the output of the following C program:

```
# include <stdio.h>
void print 1 (void)
{
    static int x = 10;
    x+ = 5;
    printf ("%d", x);
}
Void print 2 (void)
{
    static int x;
    x = 10;
    x+ = 5;
    print f ("%d", x);
}
int main ( )
{
    print 1( ); print 1( ); print 2( ); print 2 ( );
    return 0;
}
```

(a) 15, 20, 25, 30 (b) 15, 20, 15, 20

(c) 15, 15, 15, 15 (d) None of these

**Q.16** Consider the following program segment:

```
# include <stdio.h>
int main ( )
{
    char string1 [15] = "GATE, 2017";
    char string2 [15] = "GATE, 2017";
        if (string1 == string2)
        printf ("Two strings are equal ");
    else
```

```
        printf ("Two strings are unequal");
    return 0;
}
```

Which of the following is correct?

(a) Two strings are equal
(b) Two strings are unequal
(c) Compilation error
(d) Run time error

**Q.17** In Binary search tree deletion of any internal node having left and right child both, we need inorder successor of a node (deleted node). Which of the following is true about inorder successor needed in deletion operation ?

(a) Inorder successor is always either leaf node or a node with empty right child.
(b) Inorder successor maybe a 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.18** Consider the following C program execute on a singly linked list numbered from 1 to *n* containing atleast 2 nodes :

```
struct Listnode
{
    int data;
    struct Listnode *next;
} ;
void fun (struct Listnode *head)
{
    if (head == NULL || head → next ==NULL) return;
    struct Listnode *tmp = head → next;
    head → next = tmp → next;
    free (tmp);
    fun (head → next);
}
```

Which of the following represents the output of above function 'fun' ?

(a) It reverses the every 2 adjacent nodes linked list
(b) Every odd number nodes of given linked list will be deleted
(c) Every even number nodes of given linked list will be deleted
(d) It reverses the linked list and delete alternate nodes

**Q.19** Consider the following C-code :

```
main ( )
    {
        char p[ ] = "MADEEASYTEST";
        char *y = p;
        printf("%s", p + y [7] – y [10]);
    }
```

What will be the output of above C program ?

(a) MADEEASYTEST
(b) SYTEST
(c) TEST
(d) EASYTEST

**Q.20** Suppose that queue operations are implemented using stack operation. enequeue($x$) and Dequeue( ) are queue operations whereas Pop( ) and Push($x$) are stack operations.

Consider the following code:

```
Enqueue (S1, x)
{
    Push (S1, x);
}
Dequeue (S1, S2)
{
    if(! Is empty stack S2)
    return Pop (S2);
    else
    {
       while (! Is empty stack S1)
          B1 ;
    return B2 ;
    }
}
```

Fill the missing statement $B1$ and $B2$ to perform Dequeue operation correctly (here $S_1$ and $S_2$ are two stacks)

(a) Push ($S_2$, Pop ($S_1$)); and Pop $S_2$;
(b) Push ($S_1$, Pop ($S_2$)); and Pop $S_1$;
(c) Push ($S_1$, Pop ($S_2$)); and Pop $S_2$;
(d) Push ($S_2$, Pop ($S_1$)); and Pop $S_1$;

**Q.21** Consider the following program segment prints:

```
# include <stdio.h>
void fun (int *a, int *b)
{   a = b;
    *a+ = 2;
    a = b;
}
int p = 50, q = 20;
int main( )
{   fun (&p, &q);
    fun(&p, &p);
    fun(&q, &q);
    printf ("%d %d", p, q);
    return 0;
}
```

Which of the following represents the values printed by above program segments?

(a) 50, 22
(b) 52, 24
(c) 54, 20
(d) None of these

**Q.22** Which of the following represents the number of labeled binary trees with '$n$' node, which have same preorder ?

(a) $\left(\frac{^{2n}C_n}{n+1}\right)$
(b) $\left(\frac{^{2n}C_n}{n+1}\right) \times n!$
(c) $\left(\frac{2n!}{(n)! \times n!}\right)$
(d) Can not find out

**Q.23** Consider the following C-fragment where size of int is 1 B

```
int main ( )
{
    int S[6] = {10, 20, 30, 40, 50, 60};
    int *Str = (int *) (&S + 1);
    printf (% d % d, *(S + 2), *(Str – 2))'
}
```

Which of the following is correct output ?

(a) 20, 60 (b) 30, 60
(c) 40, 50 (d) 30, 50

**Q.24** Consider the following C-program:

```
#include <stdio.h>
int main ( ) {
char *arr[ ] = {"GATE", "CAT", "IES", "IAS", "PSU", "IFS"};
      call (arr);
      return 0;
}
void call (char **ptr    ) {
      char ** ptr1;
      ptr1 = (ptr+ = size of (int)) –2;
      printf("%s\n", *ptr1);
}
```

Which of the following represents the output of above program? (Assume size of int, pointer is 4B)

(a) IES (b) IAS
(c) CAT (d) PSU

**Q.25** Consider a stack implementation supports, in addition to PUSH and POP, an operation REVERSE, which reverses the order of the elements on the stack. Which of the following represents the minimum stack operations required to implement ENQUEUE and DEQUEUE operations of queue data structure respectively?

(a) 1 and 3 (b) 3 and 1
(c) 2 and 2 (d) Either (a) or (b)

**Q.26** Consider the following statements:

**$S_1$ :** Rotation operation in AVL always preserves the inorder numbering.
**$S_2$ :** The median of all element in the AVL trees is always at root or one of its two children.
**$S_3$ :** If every node in binary search tree has either 0 or 2 child, then searching time is $O(\log n)$.
**$S_4$ :** A 3-array tree is a tree in which every internal node has exactly 3 children. The number of leaf nodes in such a tree with 20 internal will be 41.

Which of above statements are true?

(a) $S_1$, $S_2$ only (b) $S_2$, $S_3$ only
(c) $S_3$, $S_4$ only (d) $S_1$, $S_4$ only

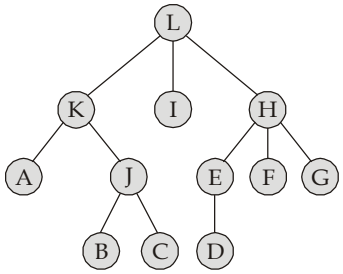
**Q.27** Consider the following function with a Binary Tree with atleast one node:

```
int path (struct node * x, int len)
{
    if (x == NULL) return (B);
    else return (A);
}
```

Assume the above function is used "to check the given binary tree has any path with specified length from root to the leaf node". Let T be a binary tree with root pointed by $x$. The function path($x$, 10) returns non-zero if there exist any path from root to leaf has a path length of 10. Otherwise return zero. Find B and A with the recursive calls of path?

(a) A is path($x \rightarrow$ left, len–1)||path($x \rightarrow$ right, len–1), B is (len = = 0)
(b) A is path($x \rightarrow$ left, len–1)||path($x \rightarrow$ right, len–1), B is (len = = 1)
(c) A is path($x \rightarrow$ left, len–1)||path($x \rightarrow$ right, len–1), B is (len = = –1)
(d) A is path($x \rightarrow$ left, len)||path($x \rightarrow$ right, len), B is (len = = 1)

**Q.28** Consider the following:




Which of the following represent Inorder and Postorder of the tree?

(a) Inorder: ABKJCLIEDHFG
Postorder: ABCJKIDEFGHL

(b) Inorder: AKBJCILDEFHG
Postorder: ABCJKIDEFGHL

(c) Inorder: AKBJCLIDEHFG
Postorder: ABCJKIDEFGHL

(d) Inorder: ABCJKILDEHFG
Postorder: ABCJKLIEDHFG

**Q.29** Consider the following functions, googly( ), doosra( ) and teesra( ). Note that, a variable has a bool type if it holds a value in {true, false}. Also, $\log_2(n)$ computes the base 2 logarithm of the input number $n$.

**FUNCTION 1**

```
bool doosra(int n)
{
    return (ceil(log2(n)) == floor(log2(n)));
}
```

**FUNCTION 2**

```
bool googly(int n)
{
    if (n == 0) return false;
    while (n! = 1)
    {
    if (n% 2! = 0) return false;
    n = n/2;
    }
    return true;
}
```

**FUNCTION 3**

```
bool teesra(int x)
{
    return x && (! (x &(x – 1)));
}
```

Which of the above functions produce the same output for a given input?

(a) Googly, Doosra
(b) Doosra, Teesra
(c) All 3 are equivalent
(d) None of these

**Q.30** Consider the following 3 programs:

**Program $P_1$ :**

```
int *g(void) {
    int x = 10;
    return (&x);
}
```

**Program $P_2$ :**

```
int *g(void) {
    int *px;
    *px = 10;
    return px;
}
```

**Program $P_3$ :**

```
int *g(void) {
    int *px;
    px = (int*) malloc (sizeof (int));
    *px = 10; free(px);
    return px;
}
```

Which of the above three functions are likely to cause problems with pointers?

(a) Only $P_2$
(b) Only $P_1$ and $P_3$
(c) Only $P_1$ and $P_2$
(d) $P_1$, $P_2$ and $P_3$

■■■■

CLASS TEST

Sl. : 01JPCS-PDS-19072023

MADE EASY
India's Best Institute for IES, GATE & PSUs

Delhi | Bhopal | Hyderabad | Jaipur | Pune | Bhubaneswar | Kolkata

**Web:** www.madeeasy.in | **E-mail:** info@madeeasy.in | **Ph:** 011-45124612

# PROGRAMMING & DATA STRUCTURES

## COMPUTER SCIENCE & IT

**Date of Test : 19/07/2023**

## ANSWER KEY ➤

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | (a) | 7. | (b) | 13. | (b) | 19. | (b) | 25. | (d) |
| 2. | (d) | 8. | (c) | 14. | (a) | 20. | (a) | 26. | (d) |
| 3. | (b) | 9. | (a) | 15. | (d) | 21. | (b) | 27. | (c) |
| 4. | (d) | 10. | (a) | 16. | (b) | 22. | (a) | 28. | (c) |
| 5. | (a) | 11. | (d) | 17. | (d) | 23. | (d) | 29. | (c) |
| 6. | (d) | 12. | (a) | 18. | (c) | 24. | (a) | 30. | (d) |

## DETAILED EXPLANATIONS

**1. (a)**

In case of full or complete binary tree,

$$\text{Minimum height } (h_{min}) = \lceil \log_2 (n+1) \rceil$$

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

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

**2. (d)**

(a) 6, 8, 4, 7, 5


After popping element 6, only 4 can be popped, hence this permutation is not possible.

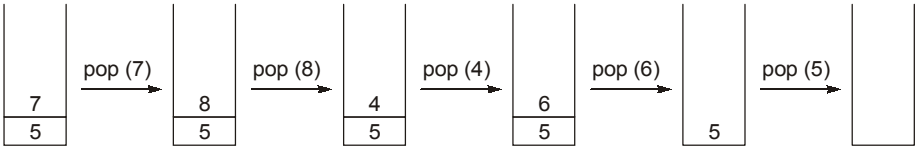
(b) 6, 4, 5, 7, 8


After performing pop operation on element 6, 4 now only element 8 can be popped.

(c) 6, 4, 7, 8, 5


After 6, 4 elements are popped, now element 7 can only be popped iff 8 has already been popped.

(d) 7, 8, 4, 6, 5




**3. (b)**

Here m represent the number of rows and n represents the number of column.

m = 2, n = 3

∗ (A[0] + 0) = A[0][0] = 1

∗ (A[1] + 0) = A[1][0] = 4

Similarly it will access all the element.

∴ 1 4 2 5 3 6 is the output printed by the program.

**4. (d)**

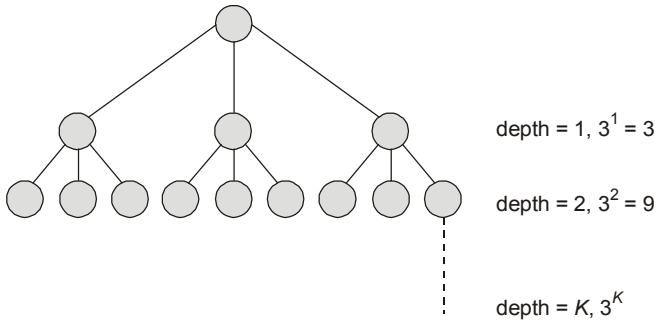
- Void pointers can't be used for dereferencing because each variable type takes different amount of memory.
- Since compiler can not know, after how many bytes is the next variables located, hence arithmetic operations can't be performed.
- Default value of extern storage class is 0.

**5. (a)**

Number of children by every node = $n$

depth of tree = $k$

**Let $n = 3$**




Hence the maximum number of leaves that '$T$' an have in $n^k$.

**6. (d)**

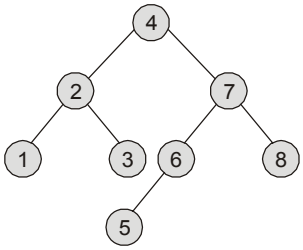
Let $n = 3$

$$r = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & -3 & -8 \\ -3 & 2 & -5 \\ -8 & -5 & 3 \end{bmatrix} \end{array}$$

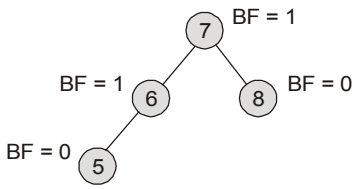
$$1 + 2 + 3 + 2 \times (-3 - 8 - 5) = -26 \neq \frac{n(n+1)}{2}$$

**7. (b)**

The tree constructed will be,



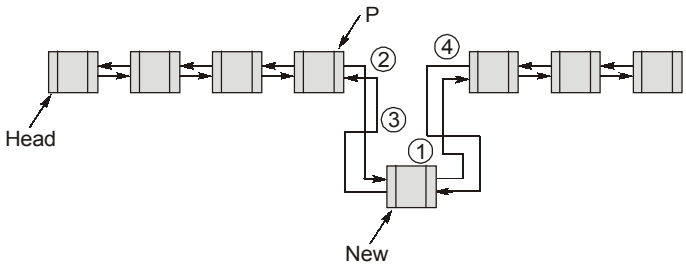

- '$T$' has 4 leaf nodes.
- Subtree rooted at node '7' satisfies the AVL property.




ΘBF =Balancing Factor

- In a heap, nodes are started from by most pointer, hence node '5' is violating the basic heap property.

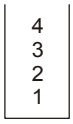
**8. (c)**



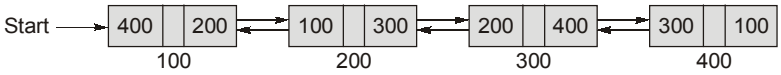

1. new → next = P → next;
2. P → next = new;
3. new → prev = P;
4. (new → next) → prev = new;

**9. (a)**

Consider a stack




The characteristic of the stack is both insertions and deletions are performed from one end.
If, it is implemented with a link lists, then both insertions and deletions are needed to be performed from the end.
Since, the linked list is a doubly circular linked list, hence the start node will have address of last node.




So, both the operations can be performed in $O(1)$ time.

**10. (a)**

Initial value are $p = -3$
$q = 2$
$r = 0$

&& has more priority than ++

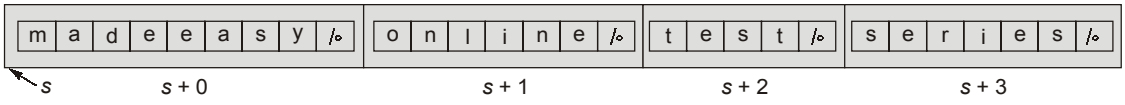
$++p = -2$
$++q = 3$

Since, both are non zero, hence expression becomes true. $r$++ need not be checked for calculating '$s$' because it's an OR operation so $s = 1$ i.e. the truth value of the expression.

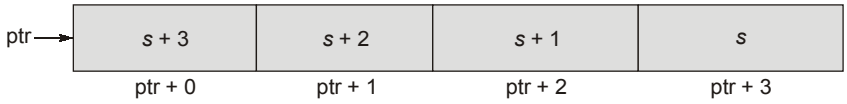
$$t = p + q + s++$$
$$= -2 + 3 + 1$$
$$= 2$$

**11. (d)**

In this problem we have an array of char pointers pointing to start of 4 strings i.e.,



We have ptr which is pointer to a pointer of type char and a variable *p* which is a pointer to a pointer of type char.




*p* = ptr; *p* ptr

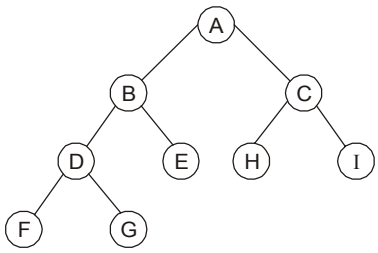
++*p*; *p* ptr+1

Printf("%s", ∗ – – ∗ ++ *p* + 3);

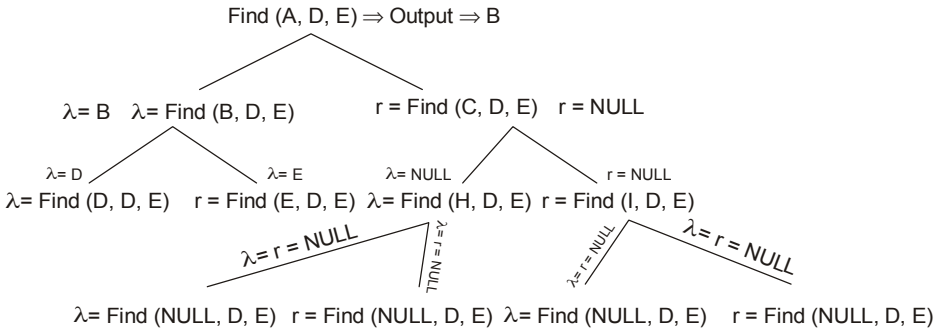
In printf statement the expression is evaluated ∗++*p* cause gets value (*s*+1) then now pre-decrement is executed and we get (*s*+1) – 1 = *s*. The indirection pointer now gets the value from the array of *s* and add 3 to the starting address. The string is printed starting from this position. Thus, the output is 'eeasy'.

**12. (a)**

Let's traverse the code on the given tree.



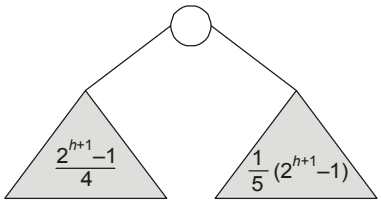

Let 'p' be D and 'q' be E.




Consider nodes 'D' and 'E', they have two ancestors node 'A' and node 'B'. 'B' is the lowest common ancestor and that is the output.

Hence, the value returned by the function is lowest common ancestor of two nodes.

**13. (b)**

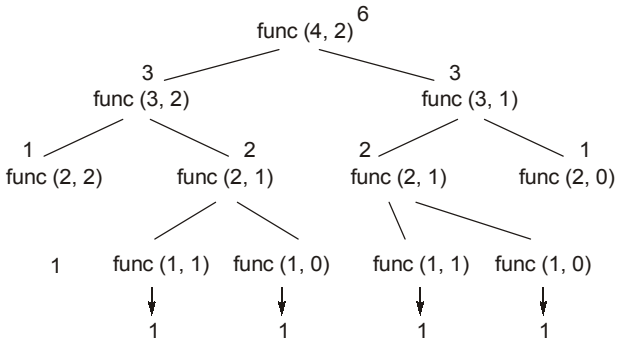



$$\text{Total} = \left(\frac{2^{h+1}-1}{4}\right) + \frac{1}{5}\left(2^{h+1}-1\right) + 1$$

$$= \frac{9}{20}\left(2^{h+1}-1\right)+1$$

**14. (a)**

Take m = 4 and n = 2




So, correct vaue of E is func (m – n, n) + func (m – 1, n – 1).

**15. (d)**

- **print 1( ):** $x$ = 10 + 5 = 15; since the variable is of static storage class, hence it will retain its value between different function calls.
- **print 1( ):** $x$ = 15 + 5 = 20; since it has retained its value 15.
- **print 2( ):** $x$ is defined again inside the function and hence will print, $x = x + 5 = 10 + 5 = 15$. Again when the function will be called, $x$ = 10 + 5 = 15. Here second time also $x$ = 10 will be there because it is not initialized at the time of definition.

Hence output 15, 20, 15, 15.

**16. (b)**

By comparing string1 and string2, we are not comparing the actual data of the string, instead the starting address of both strings will be compared.

Hence, it will print "Two strings are unequal".

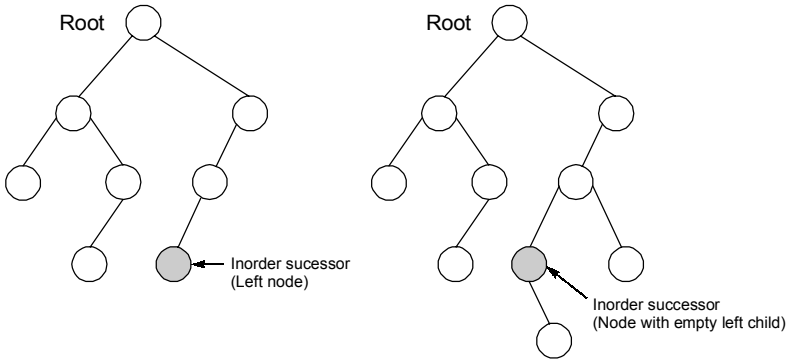
To compare the actual data of the strings,

if (∗ string1 == ∗ string2)

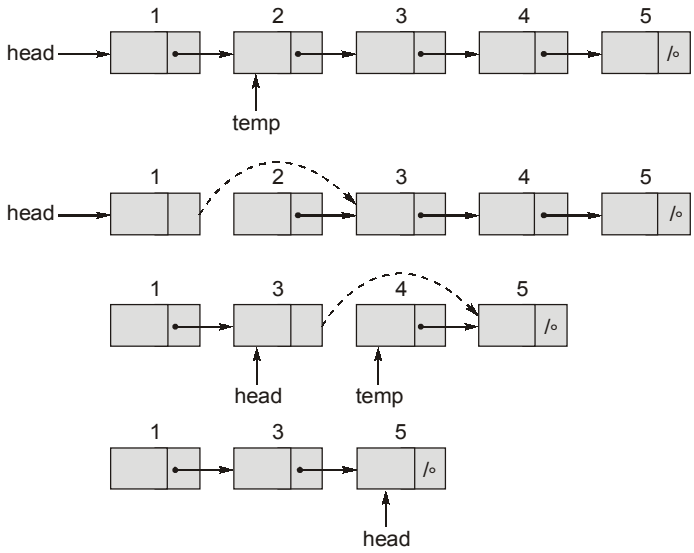
then will get the output: "Two strings are equal".

**17. (d)**

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

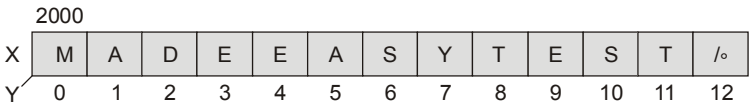


**18. (c)**




The above program deletes every even number node in the linked list (In particular second, fourth, sixth... soon nodes will be deleted)

**19. (b)**



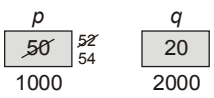

X = 2000
Y = 2000
y[7] = *y* (90 in ASCII)
y[10] = *s* (83 in ASCII)
= X + Y[7] – y[10]
= 2000 + 90 – 83 = 2007

Therefore it prints from the array starting at address; 2007 to the end i.e., "MADEEASYTEST".

**20. (a)**

If $S_2$ stack is empty and $S_1$ stack is not empty then we have to pop the element from stack $S_1$ and push that element into stack $S_2$ and return the stack $S_2$ which contain newly inserted element.

**21. (b)**

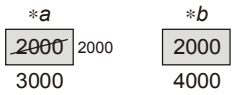



**1st** function call pass parameters as call by reference


*a* = *b* ⇒ Now a is also pointing to 2000 address.
**a*+ = 2 ⇒ **a* = **a* + 2
⇒ **a* = 20 + 2 = 22
*a* = *b* ⇒ Now a is also pointing to 2000 address.
**2nd** function call by reference


$a = b$ '$a$' will now store $b$ value which is 1000, already contain by '$a$'.

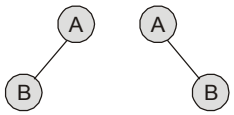
- $*a+ = 2;$
  $*a = *a + 2;$
  $*a = 50 + 2$
  $*a = 52$
- $a = b$
  '$a$' will now store $b$ value which is 1000, already contain by '$a$'.




$a = b \Rightarrow$ Now a is also pointing to 2000 address.
$*a+ = 2 \Rightarrow *a = *a + 2$
$\Rightarrow *a = 22 + 2 = 24$
$a = b \Rightarrow$ Now a is also pointing to 2000 address.
So, output will be 52 and 24.

**22. (a)**

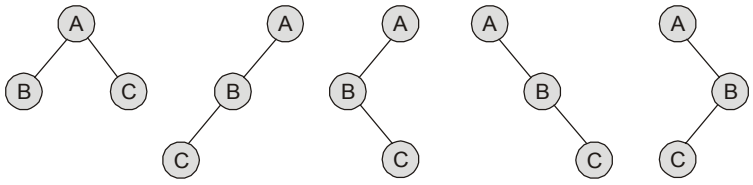
Consider 2 nodes "A", "B"
Consider preorder = A . B




only 2 Tree possible

So, $\frac{4!}{3! \times 2!} = \frac{4}{2} = 2$

Consider 3 nodes A. B. C
Consider preorder = A, B, C

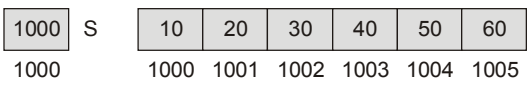



Only 5 trees present

So, $\frac{6!}{4! \times 3!} = \frac{6 \times 5}{3 \times 2} = 5$

So option (c) is true.

**23. (d)**




str 1006 $= (\text{int} *)(\&S + 1)$

$\Rightarrow = (\text{int} *)(\text{Base address of (S)} + 1 * (\text{size of (S)})$

$\Rightarrow$ = (1000 + 6 B)
$\Rightarrow$ = (1000 + 6)
$\Rightarrow$ = 1006

Now, *(S + 2) print 3rd element from start.
*(Str – 2) print *(1006 –2) = *(1004) element at address 1004 i.e. 50.

**24. (a)**

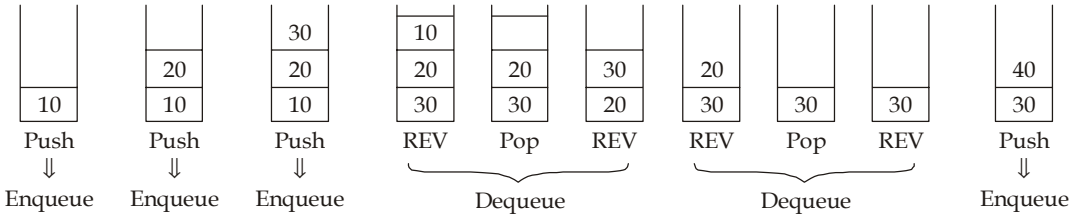
| arr[ ] = | GATE% | CAT% | IES% | IAS% | PSU% | IFS% |
|---|---|---|---|---|---|---|
| | 1000 | 1004 | 1008 | 1012 | 1016 | 1020 |

**ptr = arr $\Rightarrow$ **ptr = 1000;
*ptr1 = (ptr+ = size of (int)) [–2];
= (1000 + 4) [–2]
= [1000 + 4 × 4] [–2]
= [1016] [–2]
= [1015 – 2 × 4]
*ptr1 = [1008]
print(*ptr1) = IES

**25. (d)**

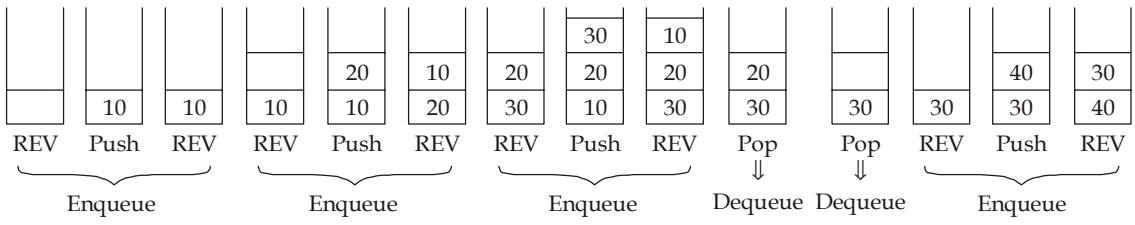
**Enqueue:** PUSH $\Rightarrow$ 1 operation
**Dequeue:** REVERSE, POP, REVERSE $\Rightarrow$ 3 operation
**Example:** Enqueue (10), Enqueue (20), Enqueue (30)
Dequeue, Dequeue, Enqueue (40)

**Queue:** ~~10~~ ~~20~~ 30 40




**Enqueue:** REV, PUSH, REV
**Dequeue:** POP
**Example:** Enqueue (10), Enqueue (20), Enqueue (30)
Dequeue, Dequeue, Enqueue (40)

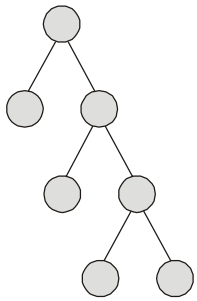
**Queue:** ~~10~~ ~~20~~ 30 40




So, either 1 Enqueue and 3 Dequeue or 3 Enqueue and 1 dequeue operation possible.

**26. (d)**

- Rotation operation in always preserves the inorder numbering so 1st is true.
- AVL tree doesnot guarantee that both left and right subtree has equal number of nodes, so statement is false.
- Consider



satisfying the property of statement 3, in this tree if element present is at last level the time complexity will be $c \times n/2 \simeq O(n)$. So $S_3$ is false.

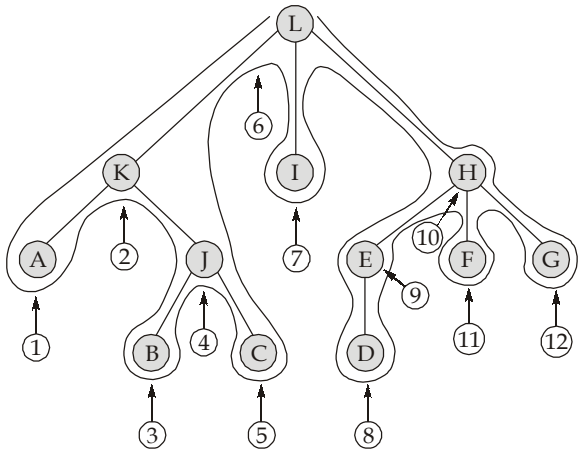
- Total nodes = 3 × internal nodes + 1
  = 3 × 20 + 1 = 61
  and 20 + 41 = 61
  (Leaf + internal = total) so $S_4$ is true.

**27. (c)**

Given function call is recursive.
Before calling any recursive call, it decrements length. So at leaf node recursive call will decrement by 1 even there exist no path.
∴ B is (len == –1)
Before traversing its child it decrements the length, whenever a length reaches to –1 and node is leaf then it implies there exist a path with given length.
∴ A is path($x \rightarrow$ left, len – 1)||path($x \rightarrow$ right, len – 1)
If one of the path returns non-zero then it recursively returns back.
So option (c) is correct.

**28. (c)**

- Inorder of tree is: Left, Root, Right (any other)




So, Inorder: AKBJCLIDEHFG

- Postorder of tree is Left, Right (any other), Root
  So, Post order is: ABCJKIDEFGHL

**29. (c)**

All the 3 functions check if a given number is a power of 2.

**Function 1:** Checks if $\log_2(n)$ of a number is an integer. If yes, it returns true, else it returns false. So function 1 checks if a given number is a power of 2.

**Function 2:** The key here is that, a number which is a power of 2 has the bit pattern 10* (1 followed by any number of zeroes). So at every step we keep checking if the number is even and keep dividing the number by 2 (right shift); if except for the most significant bit, a bit is found to be 1 (the number is odd at any point of time while right shifting), then the function 2 returns false. Else it returns true. So function 2 also checks if a given number is a power of 2.

**Function 3:** The observation is that, if a number $n$ is power of 2, then $(n-1)$ becomes the 1's complement of $n$. Hence function 3 also checks if a given number is a power of 2.

**30. (d)**

Since $P_1$ returns the address of a variable which is declared locally, $P_1$ may cause problems.

$P_2$ will cause a problem because $px$ doesn't have any address and is being dereferenced.

$P_3$ also will cause problems because even though malloc has been used to allocate the memory into the heap, free( ) has been called and returning that address is simply asking for trouble.

■■■■