CLASS TEST

S.No.:01JP_CS_30092026

MADE EASY
Leading Institute for IES, GATE & PSUs

Delhi | Bhopal | Hyderabad | Jaipur | Pune

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

Programming and Data Structures

COMPUTER SCIENCE & IT

Date of Test : 30/09/2026

ANSWER KEY ➤

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

DETAILED EXPLANATIONS

1. (a)

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

$$X + 3 = 2000 + 3 * (12) = 2036$$
$$*(X + 3) = *(2000 + 3 * (12)) = 2036$$
$$*(X + 2) + 3 = *(2000 + 2 * (12)) + 3*4 = 2036$$

2. (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.

$$x = 15 + 20 + 15 + 15 = 65$$

3. (a)

Considering, both the statements:

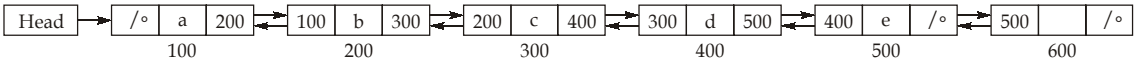
- 

$\therefore$ With a single array, two stacks can be implemented.

- Implement a queue using 2 stacks. Denote the two stacks S_1 and S_2. The enqueue operation is simply implemented as a push on S_1 which will be in O(1) time. But, while performing dequeue operation, element of S_1 will be moved to S_2, then pop from S_2 and push to S_1, each element of S_2 which takes O(n) time. It can be vice versa too i.e. enqueue in O(n) and dequeue O(1) time.

4. (a)

Consider the following doubly linked list



After the insertion, only one pointer that is the next of node 'e' is modified to 600.
Two more pointers are added in the list, but in the existing list only one pointer is modified.

© Copyright: MADE EASY

5. **(a)**

x > y ? y > z ? 1 : z > x ? 2 : 3 : y > z ? 4 : z > x ? 5 : 6;

Number of ? = 5

Number of : = 5

- If both are equal there check the condition.
- Every colon should match with just before question mark.
- Every question mark followed by colon not immediately but following.

6. **(a)**

It defines an array each element of which is a pointer to a structure of type node.

7. **(b)**

Player will output 1, player + 1 will give 2.

8. **(c)**

Initialize three pointers prev as NULL, current as head and next as NULL.
Iterate through the linked list. In loop, do following.
Before changing next of current, store next node

$$next = current \rightarrow next$$

Now change next of current, this is where actual reversing happens

$$current \rightarrow next = prev$$

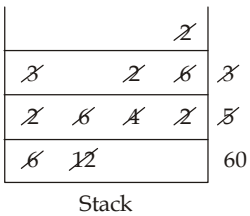
Move prev and current one step forward

$$prev = current$$
$$current = next$$

9. **(a)**

$$N(h) = N(h-1) + N(h-2) + 1$$
$$N(0) = 1$$
$$N(1) = 2$$
$$N(2) = 2 + 1 + 1 = 4$$
$$N(3) = 7$$
$$N(4) = 12$$
$$N(5) = 20$$
$$N(6) = 33$$
$$N(7) = 54$$
$$N(8) = 88$$
$$N(9) = 143$$
$$N(10) = 232$$

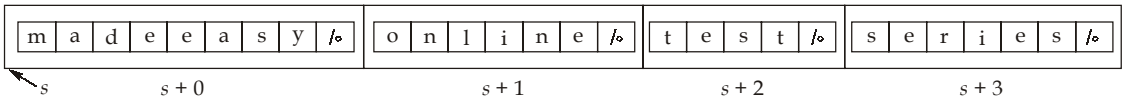
10. **(a)**



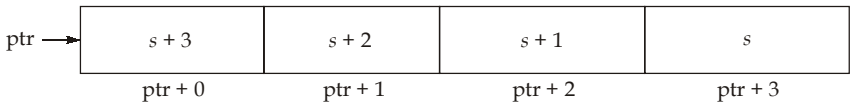
Stack

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. (b)

Implement the stack where each entity stores two values:

1. Value = Current number.
2. CurMax = maximum of current number and numbers below the current number.

To implement:

Push: If stack size is 0, add an entry with value = current number and curmax = current. If stack size >0 add an entry with value = current number and curmax = max (current number, curmax of top value on stack).

Pop: Same as normal stack.

Max: Return curmax of top entry on stack.

Every entry will be of 8 B.

After all the operation 24 B are needed.

5	6	
~~8~~	~~8~~	Max = 6
~~6~~	~~6~~	
~~7~~	~~7~~	Max = 6
6	6	
5	5	

Value Curmax

13. (c)

The given program compute the binary value of decimal number 156.

Hence, the output received will be 10011100.

© Copyright: MADE EASY

14. (a)

a	90	98	99	96	84	70
	a+0	a+1	a+2	a+3	a+4	a+5
p	a+2	a+1	a	a+3	a+4	a+5
	p+0	p+1	p+2	p+3	p+4	p+5
S	p+4	p+5	p+1	p	p+2	p+3
	S+0	S+1	S+2	S+3	S+4	S+5

Ptr | S + 2 |

$$\begin{aligned} *** (\text{ptr} + 3) - **(p + 3) &= (*(*(*(S + 3 + 2)))) - (* (*(p + 1))) \\ &= (*(*(p + 3))) - (*(a + 1)) \\ &= 96 - 98 = -2 \end{aligned}$$

15. (a)

MUL (MUL (a + 1, b), pow (b + 1))

MUL ([a + 1 * b], [b + 1 * b + 1])

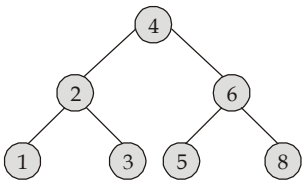
$\Rightarrow$ a + 1 * b * b + 1 * b + 1

$\Rightarrow$ 3 + 1 * 2 * 2 + 1 * 2 + 1

$\Rightarrow$ 3 + 4 + 2 + 1 = 10

16. (c)

Consider the following Binary Search Tree,



Consider 2 Scenarios :

Scenario 1: $k_l = 1; k_p = 2$

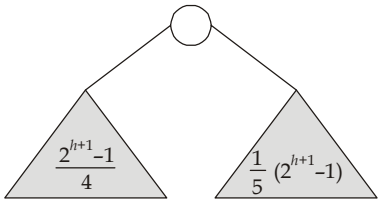
Here, k_p is the smallest key greater than k_l

Scenario 2: $k_l = 3; k_p = 2$

Here, k_p is the longest key which is smaller then k_l

Hence, either of the two is possible depending on the key.

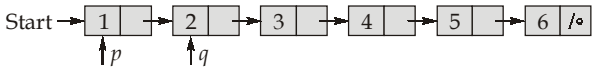
17. (b)



$$\text{Total} = \left(\frac{2^{h+1}-1}{4}\right) + \frac{1}{5}\left(2^{h+1}-1\right) + 1 = \frac{9}{20}\left(2^{h+1}-1\right) + 1$$

18. (b)

Consider the linked list.



1st function call: temp = 1

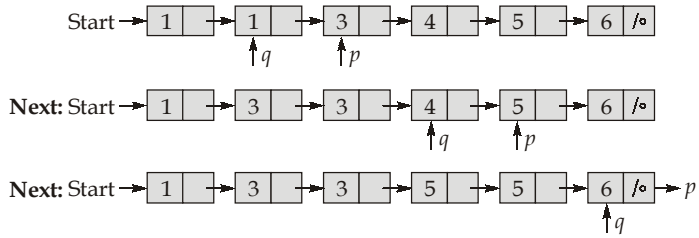
$q \rightarrow$ val = 1

$p \rightarrow$ val = $q \rightarrow$ val = 1

$q = p$, true

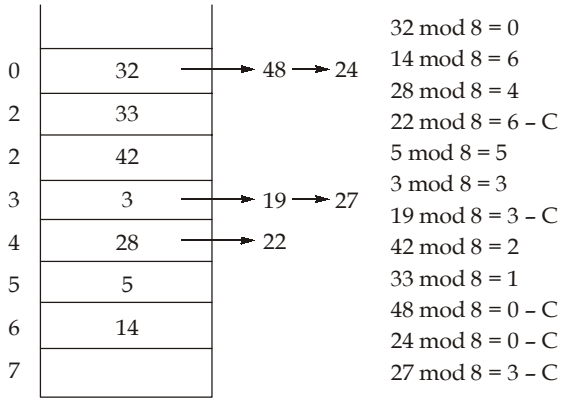
$q = p \rightarrow$ next

$p = q \rightarrow$ next



Final output: 1, 3, 3, 5, 5, 6.

19. (b)



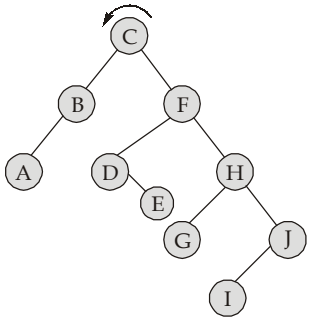
C : collision

20. (c)

Constructing BST with the help of pre-order and in-order traversals:

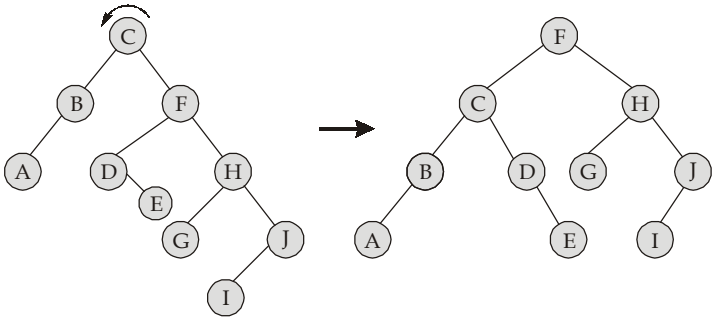
Pre-order: CBAFDEHGJI

In-order: ABCDEFGHIJ



© Copyright: MADE EASY

The tree become unbalanced at the root node. Hence applying RR rotation, we get



Hence after 1 rotation the tree travers into an AVL tree.

21. (b)

Let array be $A[lb_1.....lb_1]$ $[lb_2.....lb_2]$ $[lb_3.....lb_3]$

↓ ↓ ↓

No. of x planes n_{1r} | No. of rows in 2D array n_{2r} | No. of columns in 2D array n_{3r}

$$n_{1r} = ub_1 - lb_1 + 1$$
$$n_{2r} = ub_2 - lb_2 + 1$$
$$n_{3r} = ub_3 - lb_3 + 1$$

For row major order

$$\text{loc}[i][j][k] = \text{Base address} + [(i - lb_1) \times n_2r \times n_3r + (j - lb_2) \times n_3r + (k - lb_3)] \times c$$

where, c = Size of each element

Thus using this formula:

$$\text{loc}[16][20][2] = 400 + [(16 - 0) \times 41 \times 41 + (20 - 0) \times 41 + (2 - 0)] \times 2$$
$$= 400 + [26896 + 820 + 2] \times 2$$
$$= 400 + 55436 = 55836$$

22. (b)

Option (b) is correct answer as it returns pointer to an integer.

23. (d)

- Random access is not allowed in linked list but allowed in array.
- Insertion and deletion in array is difficult, since we need to shifts other elements, while insertion and deletion in linked list in easy (only few pointers need to be charged in worst case both take O(n) time).
- Since array are contiguous memory blocks, so large chunk of them will be loaded into the cache upon cache upon first access. Which make cache access faster for future elements of array i.e. gives better cache locality which is not possible in case of linked list.

24. (c)

Scope of a static variable is within the body.

25. (a)

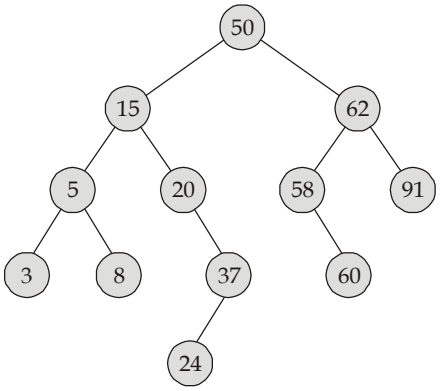
arr[ ] =	GATE%	CAT%	IES%	IAS%	PSU%	IFS%
	1000	1004	1008	1012	1016	1020

$$
\begin{aligned}
**ptr &= arr \Rightarrow **ptr = 1000;\\
*ptr1 &= (ptr+ = size\ of\ (int))\ [-2];\\
&= (1000 + 4)\ [-2]\\
&= [1000 + 4 \times 4]\ [-2]\\
&= [1016]\ [-2]\\
&= [1015 - 2 \times 4]\\
*ptr1 &= [1008]\\
print(*ptr1) &= IES
\end{aligned}
$$

26. (b)

Insertion at the end of a linked list: This code snippet inserts a new node at the end of a linked list. It takes a pointer to the head of the linked list (head_ref) and the data to be inserted (data). It traverses the list until it reaches the last node and then attaches the new node to the next pointer of the last node.

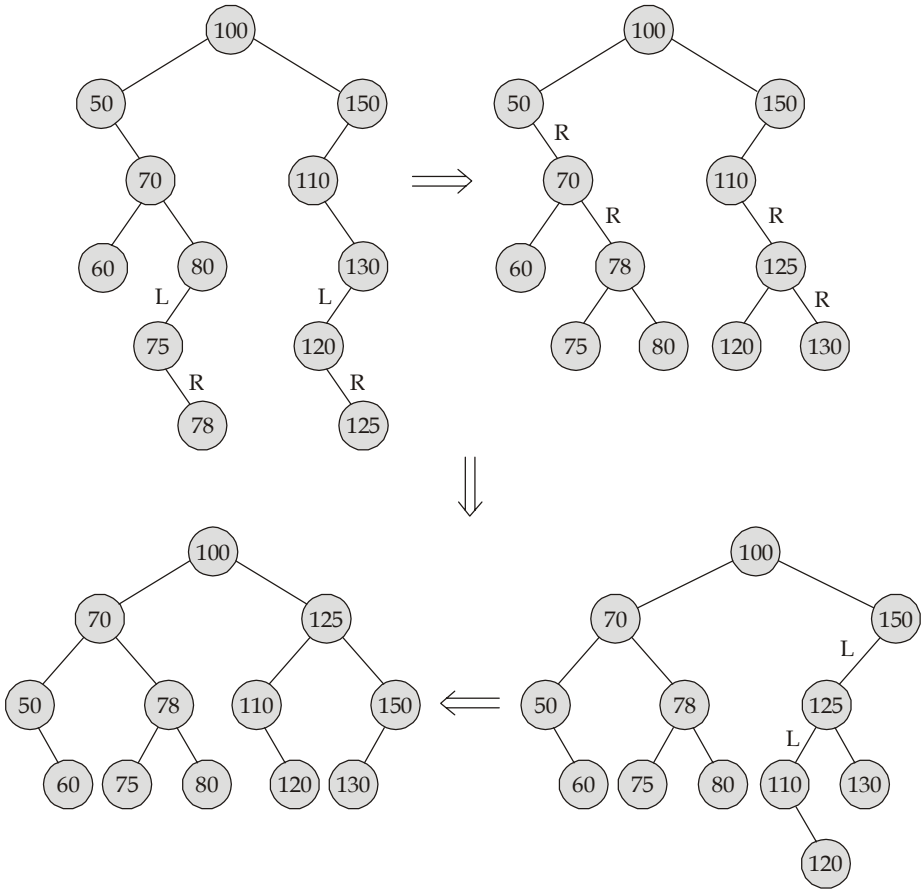
27. (b)



Here root = 50
Left sub tree nodes = 7
Right sub tree nodes = 4

© Copyright: MADE EASY

28. (c)



2 LR rotations + 2 RR rotations + 1 LL rotation

$= 2 \times 2 + 2 \times 1 + 1$

$= 4 + 2 + 1 = 7$

29. (b)

30. (c)

■■■■