CLASS TEST

S.No.: 01_SK_CS_ABCDEFGHIJKL_170825

MADE EASY
Leading Institute for IES, GATE & PSUs
Delhi | Bhopal | Hyderabad | Jaipur | Pune | Kolkata

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

PROGRAMMING & DATA STRUCTURES

COMPUTER SCIENCE & IT

Date of Test : 17/08/2025

ANSWER KEY

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

DETAILED EXPLANATIONS

1. (d)

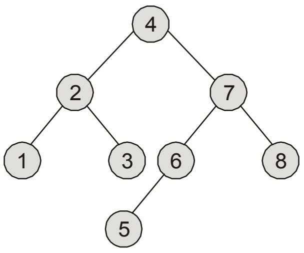
Let, $n = 3$

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

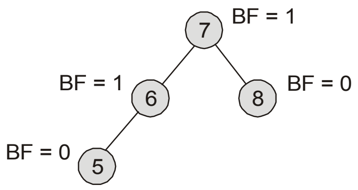
$$1 + 2 + 3 + 2 \times (-3 - 8 - 5) = -26 \neq \frac{n(n+1)}{2}$$

2. (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.

3. (c)

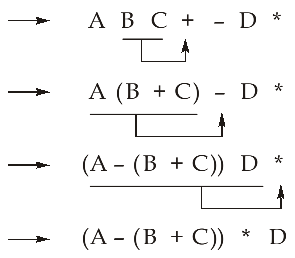
Here both *p* and *q* are holding garbage values now we are trying to modify the string constant here which is "GATE", so it will generate run time error.

4. (d)

We can solve it using the operand stack or by converting it into infix i.e.

$$[(11 + 8) * (50 / 10)] - 2 = 93$$

5. (d)



6. (d)

- False: Two case labels cannot have same value.

7. **(c)**
We have postorder traversal and the tree is BST so inorder traversal of BST is ascending order. Thus, $\theta(n \log n)$.
Using these two we can make unique tree.

8. **(c)**
First we need to traverse entire list for (p = list; $p \to$ next! = NULL; $p = p \to$ next);
when we reached at last node of list $p \to$ next! = NULL will be false and for loop terminated. Now we add node q as the next node p : $p \to$ next = q;

9. **(a)**

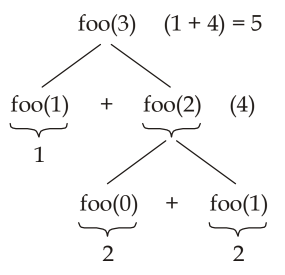
$$\underset{a}{\overset{\cancel{102}\ \ 103}{\boxed{\cancel{50}}}} \qquad\qquad \underset{b}{\overset{\cancel{51}\ \ 155}{\boxed{\cancel{50}}}}$$

$$\begin{aligned} a &= a + + + + + b \\ a &= 50 + 51 \\ a &= 101 + 1 = 102 \\ b &= b + + + + + a \\ &= 51 + 103 \\ b &= 154 + 1 = 155 \end{aligned}$$

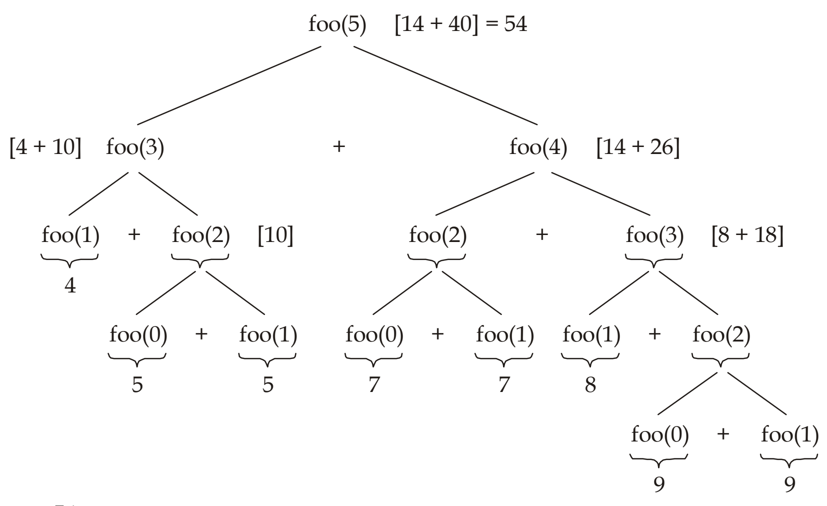
10. **(b)**
Option (b) is correct answer as it returns pointer to an integer.

11. **(b)**

$$x\ \boxed{\cancel{0}\,\cancel{1}\,\cancel{2}\,\cancel{3}\,\cancel{4}\,\cancel{5}\,\cancel{6}\,\cancel{7}\,\cancel{8}\,9}$$



foo(foo(3)) ⇒ foo(5) // x is 2



∴ 54

12. **(b)**

The given lower triangular matrix can be represented as

$$\begin{array}{c|ccccc|}
 & -6 & -5 & -4 & \ldots\ldots & +8 \\
\hline
-6 & a_{11} & & & & \\
-5 & a_{21} & a_{22} & & & \\
-4 & a_{31} & a_{32} & a_{33} & & \\
. & . & . & & & \\
. & . & . & & & \\
. & . & . & & & \\
. & . & . & & & \\
+8 & a_{81} & a_{82} & \ldots\ldots\ldots & & a_{88}
\end{array}$$

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 10th row

$= 1 + 2 + 3 + \ldots\ldots + [(i - 1) - (l_{bi}) + 1][l_{bi} \rightarrow$ lower bound of $i]$

$= 1 + 2 + 3 + \ldots\ldots (3 - (-6) + 1) = 1 + 2 + 3 + \ldots\ldots + (10)$

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

In ith 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.

Resulted address = Base address + Number of element crossed

$1000 + 63 = 1063$

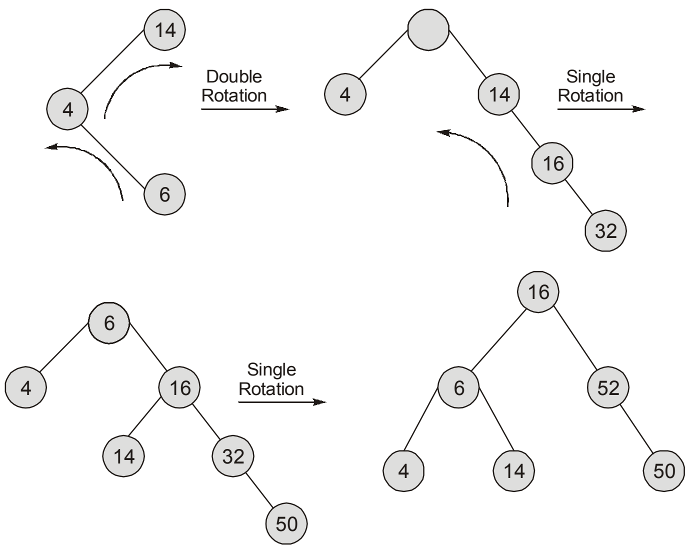
13. **(c)**

In implementation of stack using queue :

If we do some extra work in inserting the element then deletion of second element from top of stack required queue $(n - 1)$ element in another queue then again $(n - 1)$ i.e. $(n - 2)$ element to another queue so it take $O(n)$ time.

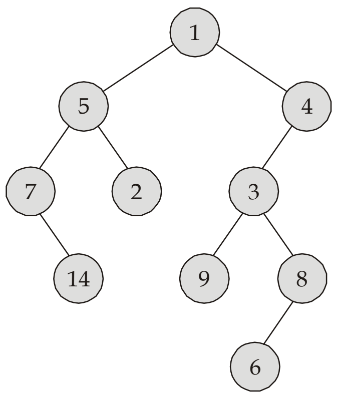
In implementation of using stack; will take $O(1)$ time to delete second element from front will take $O(1)$ time.

14. **(c)**

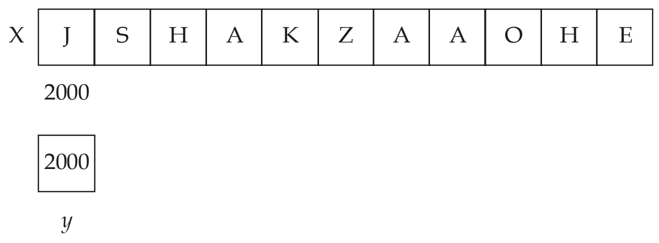


One double and two single rotations are required. So total 3 rotations.

15. **(c)**

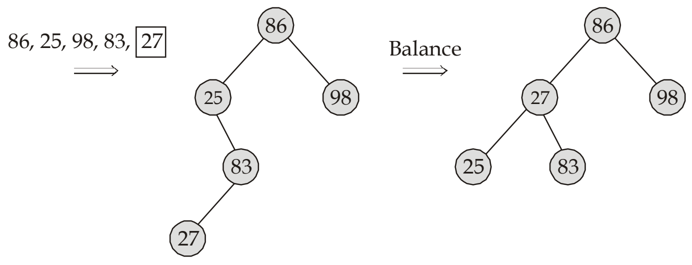


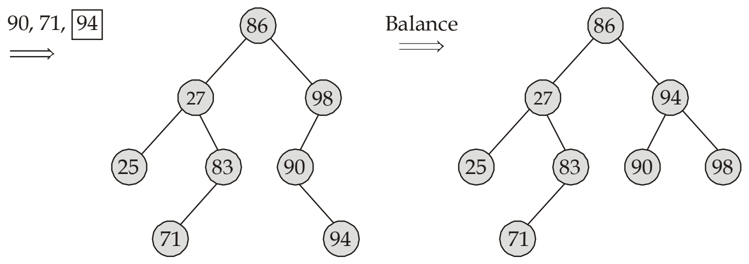
16. **(c)**



$$
\begin{aligned}
y[10] &= E\\
y[7] &= A\\
y[10] &- y[7]\\
&= \text{ASCII(E)} - \text{ASCII(A)} = 4\\
x + y[10] &- y[7] = 2004
\end{aligned}
$$

17. **(c)**





The order: 86, 25, 98, 83, 27, 90, 71, 94 will result the given AVL
Option (c) is correct.
[**Note:** Option (a) and option (b) will generate different AVL trees]

19. **(b)**

Expected number of probes in "unsuccessful search"

$$= \frac{1}{1-\alpha}$$

$$\alpha = \frac{n}{m} \text{ [Load factor]}$$

$$\Rightarrow \quad \frac{m}{m-n} = 3$$

$$\Rightarrow \quad m = 3m - 3n$$

$$2m = 3n \text{ or } \alpha = \frac{2}{3}$$

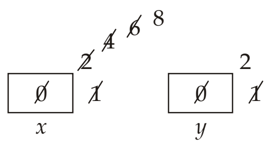
Expected number of probes in successful search

$$= \frac{1}{\alpha}\ln\left(\frac{1}{1-\alpha}\right) = \frac{3}{2}\ln 3 = 1.647$$

20. **(c)**

S_2 : True. If it is not then this node with 1 child (non-leaf) will have height imbalance problem.
S_1 : True. Node with two children gets replaced either by a successor or a predecessor.

21. **(d)**

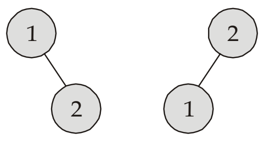


$z = \cancel{0}\ \cancel{1}\ \cancel{2}\ \cancel{3}\ \cancel{4}\ 5$

82

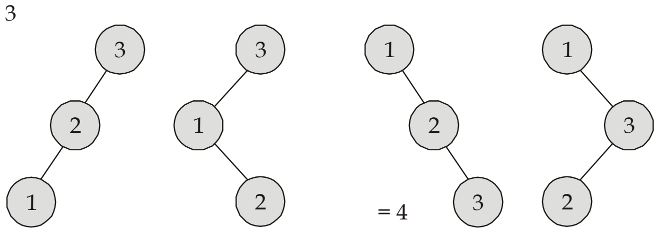
22. **(d)**

Suppose $n = 2$

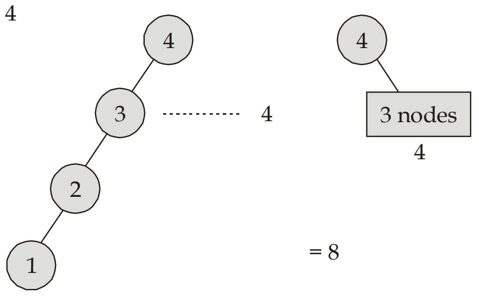


$= 2$

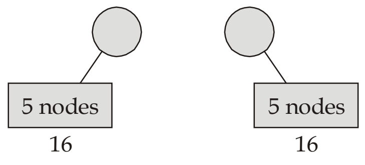
$n = 3$



$n = 4$



Similarly, $n = 6$



$= 16 + 16 = 32$

23. (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	$S + 2$

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

24. (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	

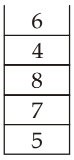
25. (c)

Applying recursion, traverse the linked list till the last element, hence at every step, the element of the linked list will be stored in the stack. While coming back, print the elements, hence the elements will be start printing from the end.

```
if (!head) return;
    printlist (head → next);
    printf("%d", head → data);
```

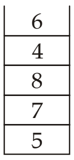
26. (d)

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



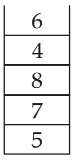
After popping element 6, only 4 can be popped, hence this permutation is not possible.

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



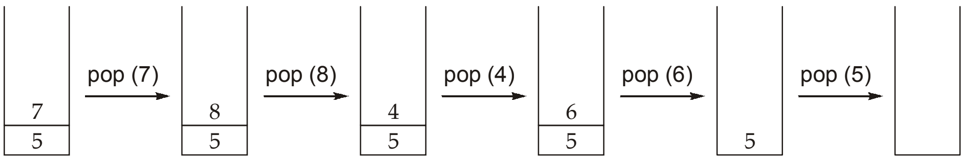
After performing pop operation on element 6, 4 now only element 8 can be popped.

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

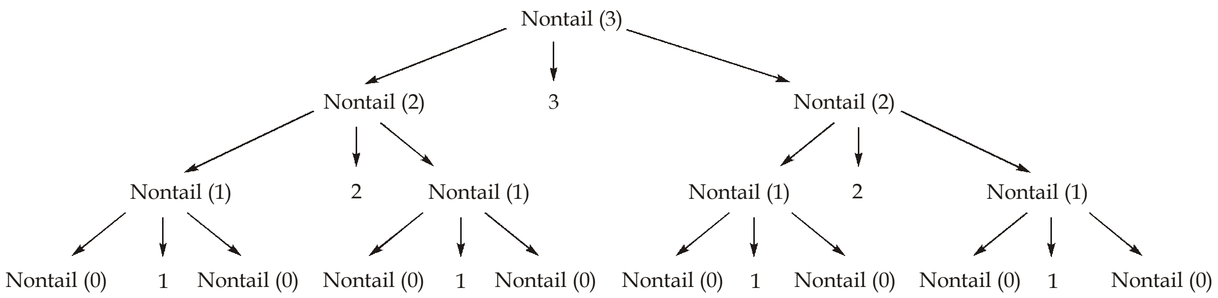


After 6, 4 elements are popped, now element 7 can only be popped iff 8 has already been popped.

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



27. (a)



Output: 1 2 1 3 1 2 1

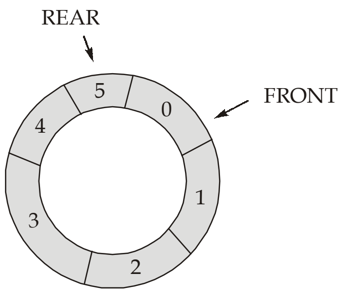
28. (b)

Queue will be empty when both REAR and FRONT will point to the same location.

i.e. REAR = FRONT.

Queue will be full when (REAR + 1) mod n = FRONT.

Example:

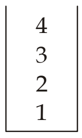


here FRONT = 0 and REAR = 5

FRONT = (5 + 1) mod 6 = 0

29. (a)

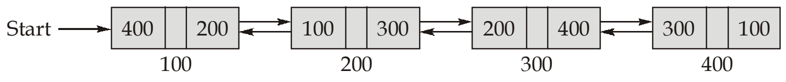
Consider a stack



The characteristic of the stack is both insertions and deletions are performed from one end.

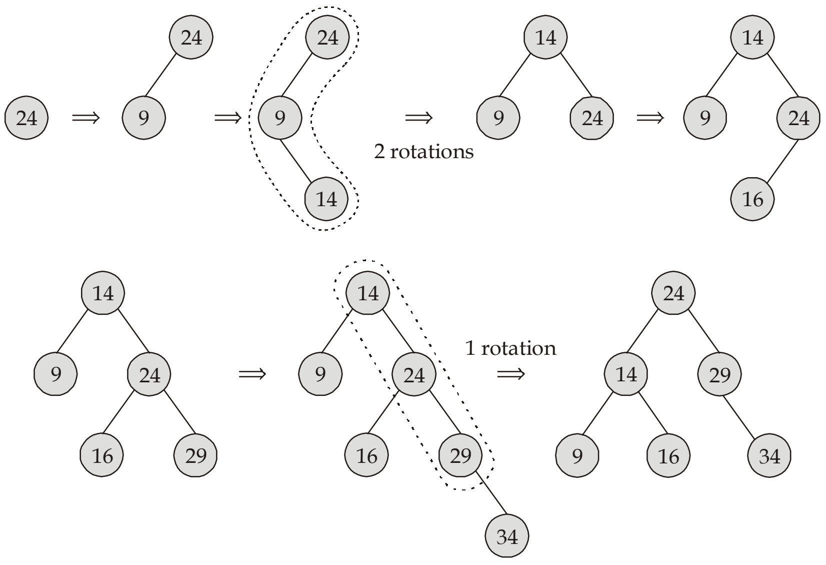
If, it is implemented with a link lists, then both insertions and deletions are needed to be performed from the end.

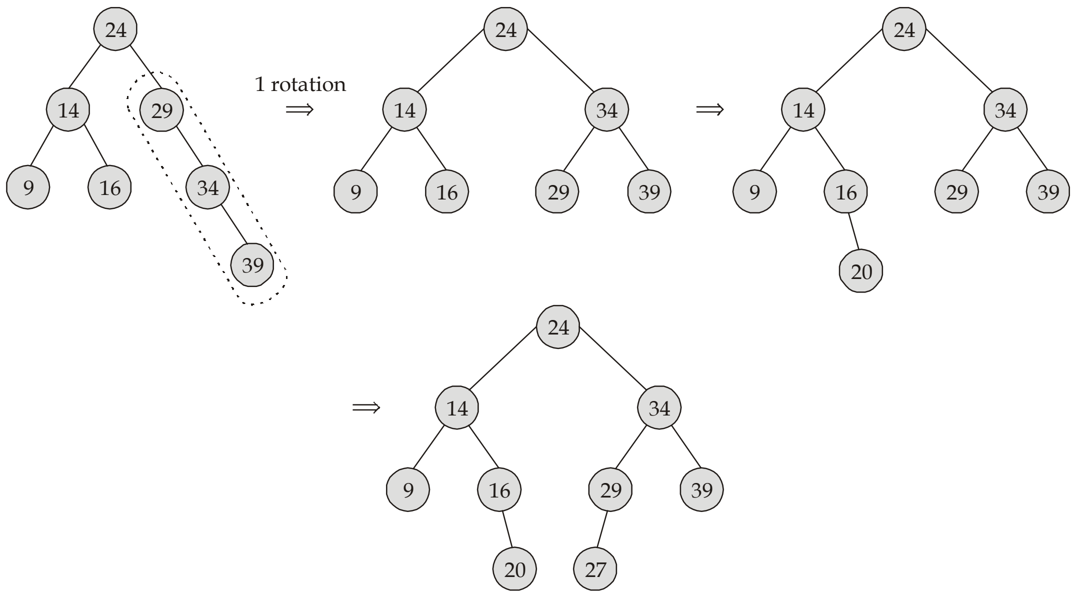
Since, the linked list is a doubly circular linked list, hence the start node will have address of last node.



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

30. (c)




1 rotation
⟹
⟹
⟹
24
14
29
34
9
16
39
20
27