CLASS TEST

S.No.:01SKCS_ABCDEFGHIJK_3108

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

Delhi | Bhopal | Hyderabad | Jaipur | Pune | Kolkata

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

Programming and Data Structures

COMPUTER SCIENCE & IT

Date of Test : 31/08/2024

ANSWER KEY ➤

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

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:

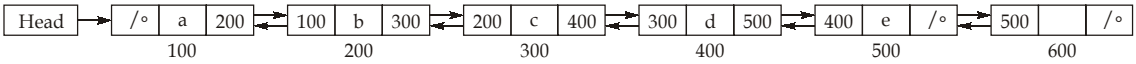
- 

$\therefore$ 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

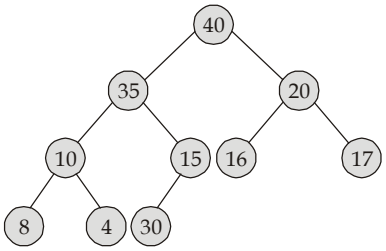


After the insertion, only one pointer that is the next of node 'e' is modified to 600.
Two more pointers are added in the list, but in the existing list only one pointer is modified.

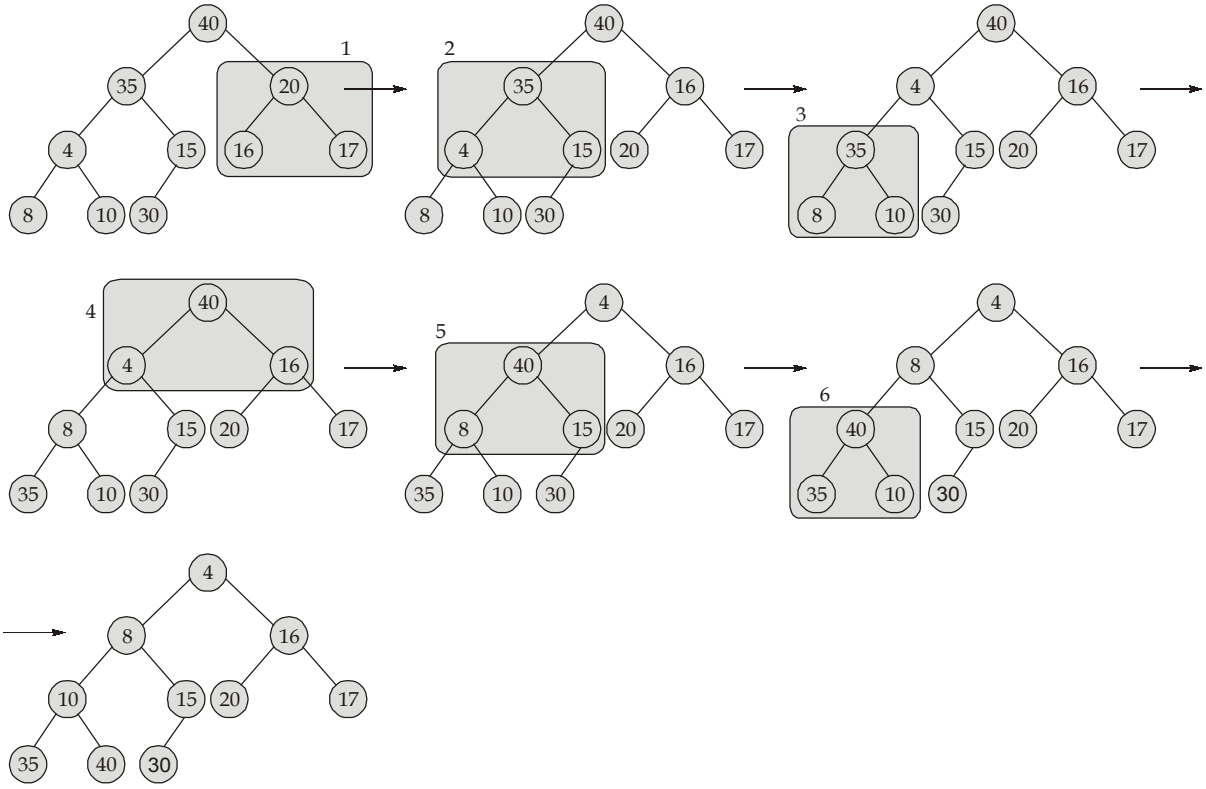
5. (d)

40	35	20	10	15	16	17	8	4	30

Constructing the Binary Tree,



In Build-heap method, it is assumed that the leaf nodes are satisfying heap property and heapify procedure starts from second last level.



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

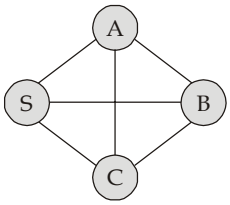
9. (a)

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

10. (a)

$9! = 362880$

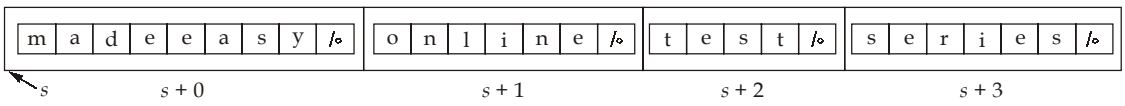
A complete graph has each vertex connected to every other vertex.
So 10 vertex complete graph with starting node S will contain (10 – 1)! BFS orderings.
Example:



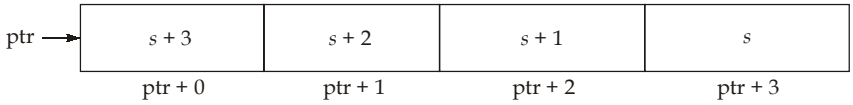
S A B C
S B A C
S B C A
S A C B
S C A B
S C B A

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.

Value	Curmax	
5	6	
~~8~~	~~8~~	Max = 6
~~6~~	~~6~~	
~~7~~	~~7~~	Max = 6
6	6	
5	5	

13. (c)

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

Hence, the output received will be 10011100.

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])

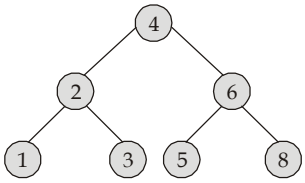
$\Rightarrow$ $a + 1 * b * b + 1 * b + 1$

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

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

16. (c)

Consider the following Binary Search Tree,



Consider 2 Scenarios :

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

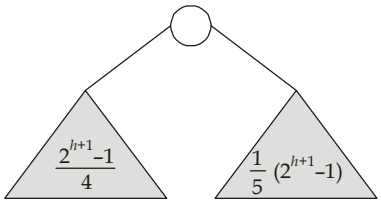
Here, k_p is the smallest key greater than k_l

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)

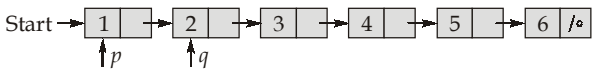


$$\text{Total} = \left(\frac{2^{h+1}-1}{4}\right) + \frac{1}{5}\left(2^{h+1}-1\right) + 1$$

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

18. (b)

Consider the linked list.



1st function call: temp = 1

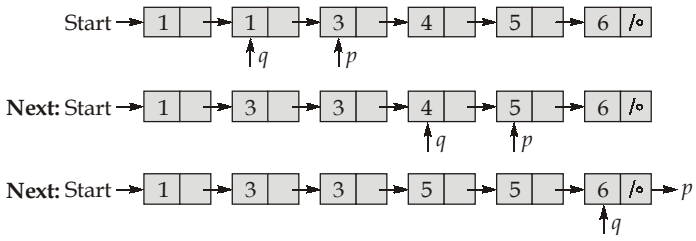
$q \rightarrow$ val = 1

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

$q = p$, true

$q = p \rightarrow$ next

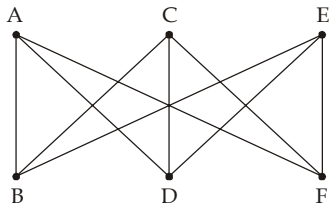
$p = q \rightarrow$ next



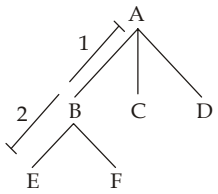
Final output: 1, 3, 3, 5, 5, 6.

19. (c)

Since bipartite graph can be grouped into 2 group of vertices i.e., $k_{3,3}$.



Take start vertex is A.



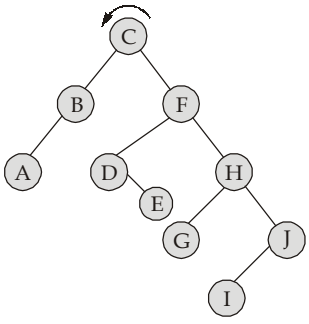
Max height of above BFS traversal is 2.

20. (c)

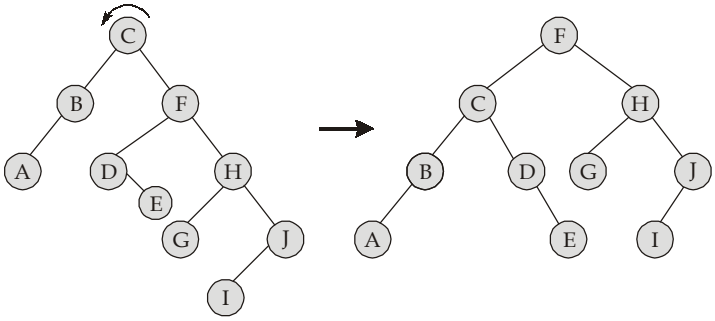
Constructing BST with the help of pre-order and in-order traversals:

Pre-order: CBAFDEHGJI

In-order: ABCDEFGHIJ



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)

Let's take $p = 16$

Now when p will be passed to function

- return (val (16/2) * log 16
  - return (val(8/2) *log 8)
    - return(val(4/2) * log 4
      - now val(2) will return 1;

so we will obtain log 16 *log 8 * log 4 * log 2 which is equal to 4*3*2*1 = 4!

So it returns (log (p))!

24. (a)

- Bitwise AND (&) in C takes two operands and does AND every bit of two numbers. The result of AND is 1 only if both bits are 1.
- Here (k&0) will return 0 always because one operand is 0.

Therefore this code will count number of 1's only.

25. (a)

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

```
**ptr  = arr ⇒ **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
```

26. (a)

Minimum cost spanning tree: (A, B), (B, C), (B, E), (E, D) will make every city reachable from one to any other, and has total cost of 21.

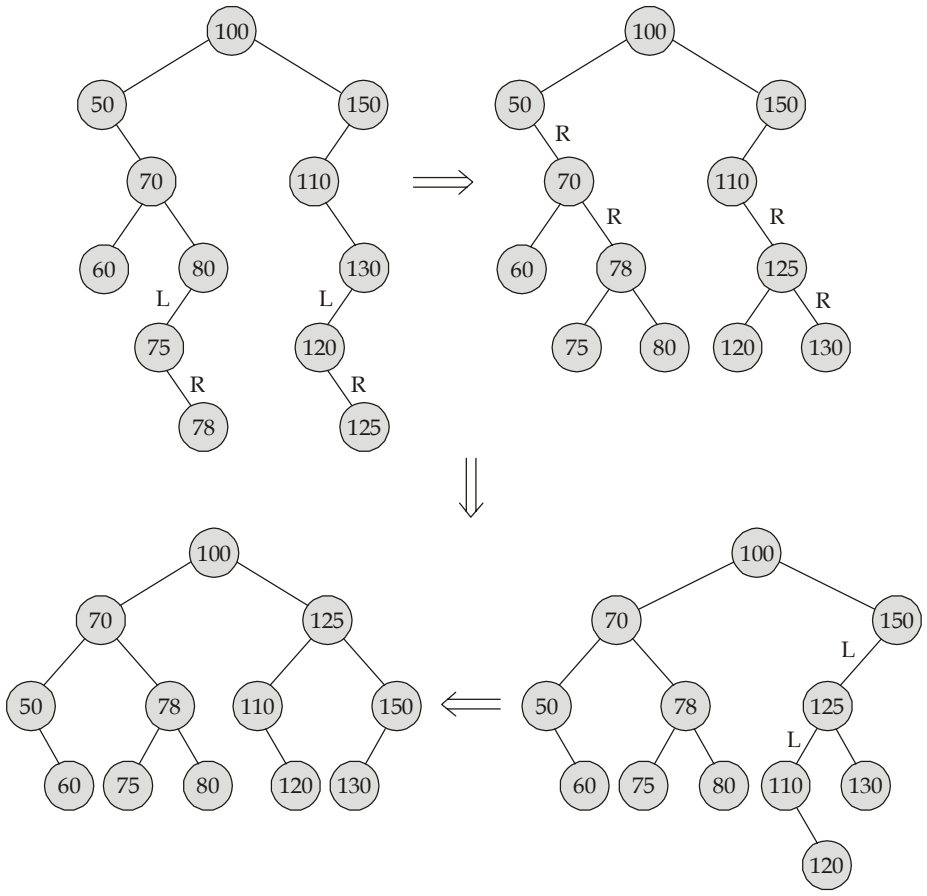
27. (c)

Create a sub graph which is a complete graph using 6 vertices.

$$\text{Edges used} = \frac{6(6-1)}{2} = 15$$

So now, 1 complete sub graph of 6 vertices and 9 other individual vertices are there.
So total 10 connected components can be created.

28. (c)



2 LR rotations + 2 RR rotations + 1 LL rotation

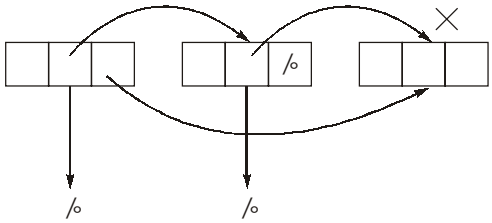
$= 2 \times 2 + 2 \times 1 + 1$

$= 4 + 2 + 1 = 7$

29. (b)

(d) Wrong. Linked list with 1 node has 2 NULL pointers.

(b) Wrong. Third last node pointer will also be updated.



30. (b)

S_1: In a max-heap tree, the minimum value will always lie in one the leaf nodes.

S_2: Worst cast time complexity of searching an element in a binary search tree is $O(n)$. So, S_2 is false.

S_3: In an AVL tree, the smallest element can be found in $O(\log n)$ time.

S_4: Stack will be used to implement depth first tree traversal. So, S_4 is false.

■■■■