MADE EASY

India's Best Institute for IES, GATE & PSUs

Important Questions *for* GATE 2022

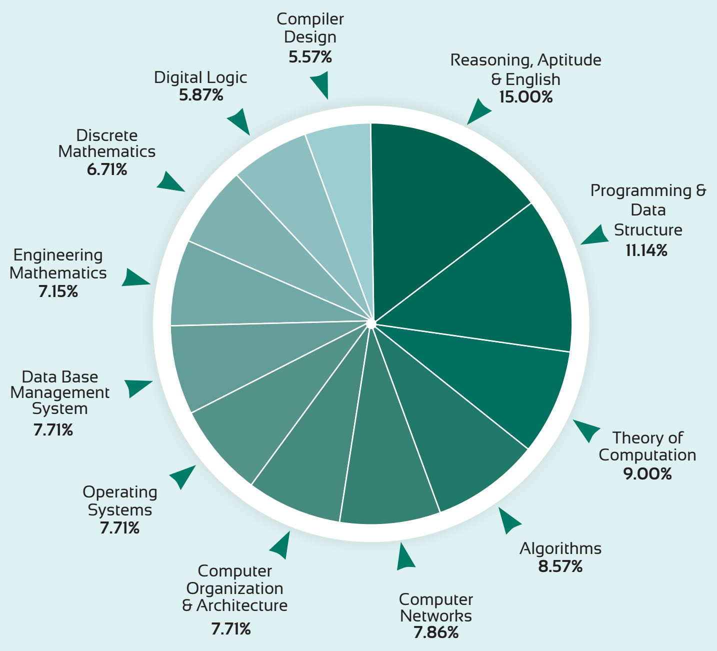
COMPUTER SCIENCE & IT

Day 2 of 8

Q.26 - Q.50 (Out of 200 Questions)

Programming and Data Structure

SUBJECT-WISE WEIGHTAGE ANALYSIS OF GATE SYLLABUS



Subject	Average % (last 5 yrs)
Reasoning, Aptitude & English	15.00%
Programming & Data Structure	11.14%
Theory of Computation	9.00%
Algorithms	8.57%
Computer Networks	7.86%
Operating Systems	7.71%
Computer Organization & Architecture	7.71%
Data Base Management System	7.71%
Engineering Mathematics	7.15%
Discrete Mathematics	6.71%
Digital Logic	5.87%
Compiler Design	5.57%
Total	**100%**

Programming and Data Structure

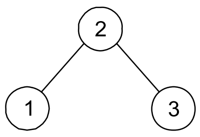
Q.26 In delete operation of binary search tree, we need inorder successor (or predecessor) of a node when a node to be deleted where it has both left and right child. Which of the following is true about inorder successor needed in delete operation?

(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.27 Consider the following program:

```
void find (struct Node *node)
{
    struct Node *ptr, *q;
    if (node == NULL) return;
    find (node → left);
    find (node → right);
    ptr = node → left;
    node → left = newNode (node → data);
    node → left → left = ptr;
}
```

If the root of following tree is passed to the above function, what is the level order traversal of output tree produced by above function? (newNode is a function which creates new node)



(a) 2 2 3 3 1 1
(b) 2 2 3 1 3 1
(c) 2 3 2 3 1 1
(d) 2 3 2 3 2 1

Q.28 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 & 01) x ++;
       n>>=1;
   }
   return x;
}
```

(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'.

Q.29 Consider the AVL tree T in which left subtree contains half of the maximum number of nodes possible in the balanced AVL tree of height h and right subtree consists of one 3rd 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) $\frac{5}{6}\left(2^{h+1}-1\right)-1$
(b) $\frac{5}{6}\left(2^{h+1}-1\right)+1$
(c) $\frac{3}{2}\left(2^{h+1}+1\right)$
(d) $\frac{3}{2}\left(2^{h+1}+1\right)-1$

Q.30 Consider the following program:

```
variable l;
procedure My (K: integer)
begin
    K = K + 1;
    Print (K);
end
procedure R( )
var l;
begin
    l = 5;
    My (l);
    print (l);
end;
begin
    l = 3;
    My (l);
    R( );
    print (l);
end
```

Find the output produced by above program using dynamic scoping, and all functions uses call by value.
(a) 4, 6, 6, 4
(b) 4, 6, 5, 3
(c) 4, 5, 6, 4
(d) 3, 6, 6, 3

Q.31 The key 14, 4, 6, 16, 32, 50 in the order are inserted into an initially empty AVL tree. Find total number of rotations to make AVL with the given keys. Assume "single rotation = 1 rotation" and "double rotation = 1 rotation".

Q.32 Assume a matrix A [–5... + 5] [5... 50] is stored in a linear in row major order. Base address of A is 0 and each element takes 1 word. Find the location of A[–2] [50].

Q.33 What does the following fragment of C program print?
char x[ ] = "JSHAKZAAOHE";
char $*y = x$;
printf("%s", $x + y$ [10] – y[7]);

(a) Prints the entire string
(b) Prints only "AKZAAOHE"
(c) Prints only "KZAAOHE"
(d) Prints only "AAOHE"

Q.34 Consider an expression: $(j + ((i - j)\ \&\ -(i < j)))$

Which of the following is true about the given expression, where i, j are integers?

(a) Finds the maximum of two integers i and j
(b) Finds the minimum of two integers i and j
(c) Finds the G.C.D of two integers i and j
(d) Finds the L.C.M of two integers i and j

Q.35 Consider the following structure for creating nodes of a double linked list.

```
struct node
{
    int data;
    struct node * x1 ; /* left pointer */
    struct node * y1 ; /* right pointer */
};
```

Let us assume P be the node in the existing linked list. Which of the following is true about the following code C_1?

$$\text{Code } C_1 : \boxed{\begin{array}{l}(P \to x_1) \to y_1 = P \to y_1\,; \\ (P \to y_1) \to x_1 = P \to x_1\,;\end{array}}$$

Assume memory is made free automatically by the compiler after deletion.
(a) It deletes the node to the right of P.
(b) It deletes the node to the left of P.
(c) It deletes the node P from the linked list.
(d) None of the above

Q.36 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) $T_i == \left(\frac{P}{Q} \times i + 1\right)$

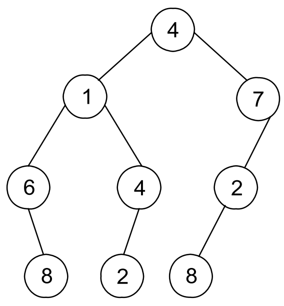
(c) $T_i == \left(\frac{P}{Q} \times (i - 1) - 1\right)$

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

Q.37 Consider the following C program:

```
struct treenode
{
    int data;
    struct treenode *left;
    struct treenode *right;
}
int gate2015 (struct treenode *root)
{
    int gate;
    if (root ==NULL) return 0;
    gate2015 (root → left);
    gate2015 (root → right);
    if (root → data % 2==1)
    {
        root → data -=1;
        gate ++;
    }
    return gate;
}
```

If the above code is executed on the following tree then what is the sum of internal nodes after the execution?



Q.38 Consider the following C function, where size represent number of elements in an array:

```
int Random (int a[ ], int size) {
        int max1 = 0,  min1 = 0, max2 = 0, start = 0, end = 0, s = 0;
for (int i = 0; i < size; i++) {
        max2 = max2 + a[i];
        if (max1 < max2) {
              max1 = max2;
              start = s;
              end = i;
        }
        if (max2 < 0) {
              max2 = 0;
              s = i + 1;
        }
}
return max1;
}
```

The output return by above function "Random" is ________.

(a) Size of maximum possible sum of array.
(b) Size of largest sum of contiguous sub-array.
(c) Maximum element in any sub-array a[ ].
(d) Sum of all the elements in the array a[ ].

Q.39 Consider the following C-program:

```
#include <stdio.h>
int value (int *x) {
        static int count;
while (*x) {
        count = count + (*x & 1);
        *x >> = 1;
}
return count;
}
int main ( ) {
}
        int a[ ] = {3, 5, 6, 4};
        int y = 0, z = 0;
        for (; y < size of (a)/size of (int); y++)
              z = a[y] + value (&a[y]);
}
```

The value of z at the end of program is ________.

Q.40 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)

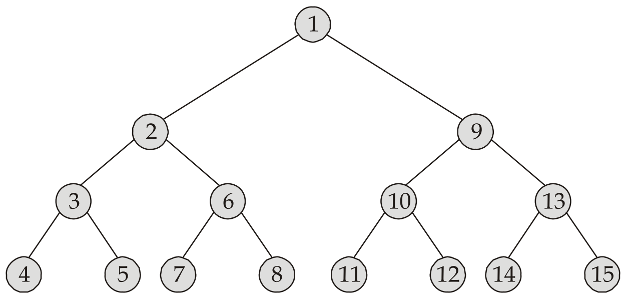
Q.41 Which of the following is true?

(a) In breadth first search of an undirected graph there are no back edge and no forward edge.
(b) In breadth first search of an directed graph there are no back edge and no forward edge.
(c) In breadth first search of an undirected graph, for each back edge (u, v), we have $0 \le v.d \le u.d$.
(d) Both (b) and (c)

Q.42 A d-ary heap is like a binary heap, but instead of two children, nodes have 'd' children. A d-ary heap can be represented in a 1-dimensional array as follows. The root is kept in A[1], its d children are kept in order in A[2] through A[d + 1] their children are kept in order in A[d + 2] through A[$d^2 + d + 1$] and so on. What index does maps the j^{th} child for ($1 \le j \le d$) of i^{th} index node?

(a) $d(i - 1) + 1$ (b) $d(i - 1) + j + 1$
(c) $d(i - 1) + j$ (d) $(d \times i) + j + 1$

Q.43 Consider a binary min heap given below containing integer in [1, 15]. The maximum number of node movement on 5 successive removal of element are ________.



Q.44 You're entrusted with the task of deleting a node in a singly linked list, whose data field is 'x'. Note that, the node which is to be deleted can be at any arbitrary position in the linked list. Consider the following scenarios.

S_1 : You're only provided with a pointer to the node which is to be deleted in the linked list.
S_2 : You're only provided with a pointer to the starting node of the linked list.

Which of the following options is correct?

(a) In both the scenarios, deletion is possible for all inputs, and deletion will be more efficient in S_1 than S_2.
(b) In both the scenarios, deletion is possible for all inputs, and deletion will be more efficient in S_2 than S_1.
(c) In S_1, deletion is not possible in certain cases; but in S_2, deletion is possible for all inputs.
(d) In S_2, deletion is not possible in certain cases; but in S_1, deletion is possible for all cases.

Q.45 Consider the following code snippet called 'Program X':

```
void f(int n)
{
    if (n <= 1) printf ("%d", n);
    else
    {
    f(n/3);
    printf("%d", n% 3);
    }
}
```

Which of the following implementations will produce the same output for $f(1023)$ as the above code?

Program P_1:

```
void f(int n)
{
    if (n/3)  {
    f(n/3);
    }
    printf("%d", n% 3);
}
```

Program P_2:

```
void f(int n)
{
    if (n <= 1) printf("%d", n);
    else
    {
    printf("%d", n% 3);
    f(n/3);
    }
}
```

(a) Both P_1 and P_2
(b) Only P_1
(c) Only P_2
(d) None of these

Q.46 Consider a hash table N slots. It is given that the collision resolution technique used is chaining. Assuming simple uniform hashing, what is the probability that the last k slots are unfilled after the first 'r' insertions?

(a) $\left(1-\frac{N}{k}\right)^r$ (b) $\left(1-\frac{k}{N}\right)^r$

(c) $\left(1+\frac{N}{k}\right)^{r-1}$ (d) $\left(1-\frac{k}{N}\right)^{r-1}$

Q.47 Consider the following program:

```
void MadeEasy (int n)
{
    printf("*");
    if (n > 1)
    {
        MadeEasy (n/4);
        MadeEasy (n/4);
        MadeEasy (n/4);
        MadeEasy (n/4);
    }
}
```

Let X be the number of asterisks printed by the above function when n = 1024. Then the value of X will be ________. (Hint: Take n as a power of 4)

Q.48 Consider the integer array A[1 ..... 100, 1 ..... 100] in which the elements are stored in Z representation. An example of a 5 × 5 array in Z representation is shown below:

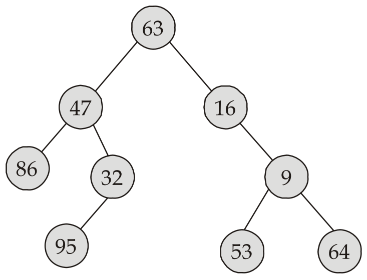
$$\begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 2 & & & & a_{24} & \\ 3 & & & a_{33} & & \\ 4 & & a_{42} & & & \\ 5 & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{array}$$

If the base address of A is starting from 1000 onwards, size of each element is 1 bytes and A is stored in Row Major Order, then the address corresponding to A[100] [55] is ________.

Q.49 Consider the following C program:

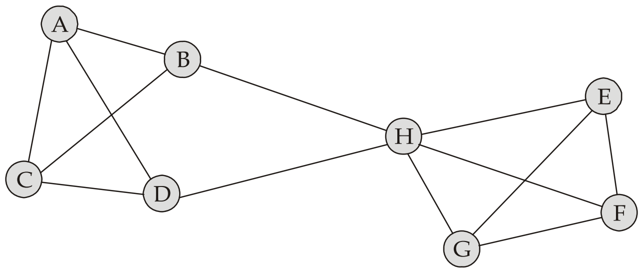
```
struct  treenode
{
    int data;
    struct  treenode *left;
    struct  treenode *right;
} ;
void Even (struct treenode *root)
{
    If (root ==NULL) return;
    Even (root → right);
    Even (root → left);
    if (root → data %2 = 0)
            (root → data) / = 2;
    else
    {
        int Sum = 0;
        if (root → left ! = NULL)  Sum + = root → left → data;
        if (root → right ! = NULL) Sum + = root → right → data;
        root → data + = Sum;
    }
    printf (“%d”, root → data);
}
```

If root of the below tree is passed as a parameter to the above function then the sum of even output values produced by the above code is _________.



Multiple Select Question (MSQ)

Q.50 Consider the following undirected graph with 8 nodes:
Now consider the following traversals made on the graph



Which of the following is/are valid Breadth First traversals?

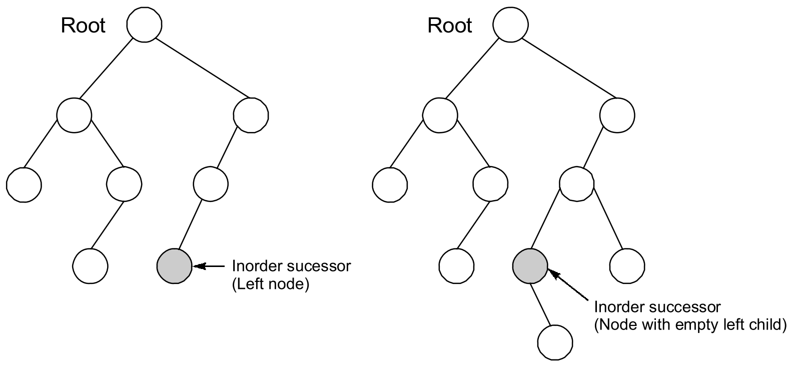
(a) ABCDHEFG
(b) ACBDHFGE
(c) ADBCHGFE
(d) HBDEGACF

■■■■

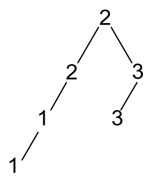
Detailed Explanations

26. **(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.



27. **(b)**



Tree after execution of above code.
Level order traversal is 2 2 3 1 3 1.

28. **(a)**

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

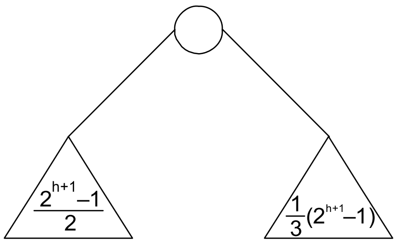
```
while (n! = 0)
{
    if (n & 01)  x ++;       /* performs bit wise AND operator and if condition is satisfied if
result                         contains atleast one 1.

    n>> = 1
}
```

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

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

29. **(b)**

Maximum nodes in AVL tree of height $h = 2^{h+1} - 1$

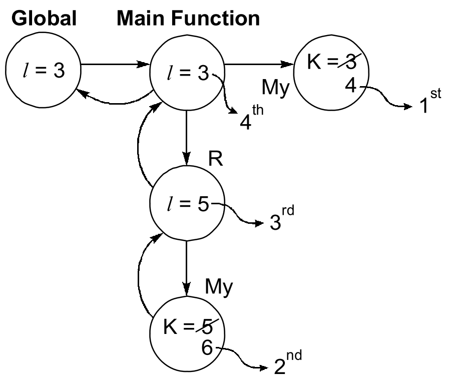


$$\therefore \quad \text{Total} = \left(\frac{2^{h+1}-1}{2}\right) + \frac{1}{3}\left(2^{h+1}-1\right) + 1$$

$$= \frac{5}{6}\left(2^{h+1}-1\right) + 1$$

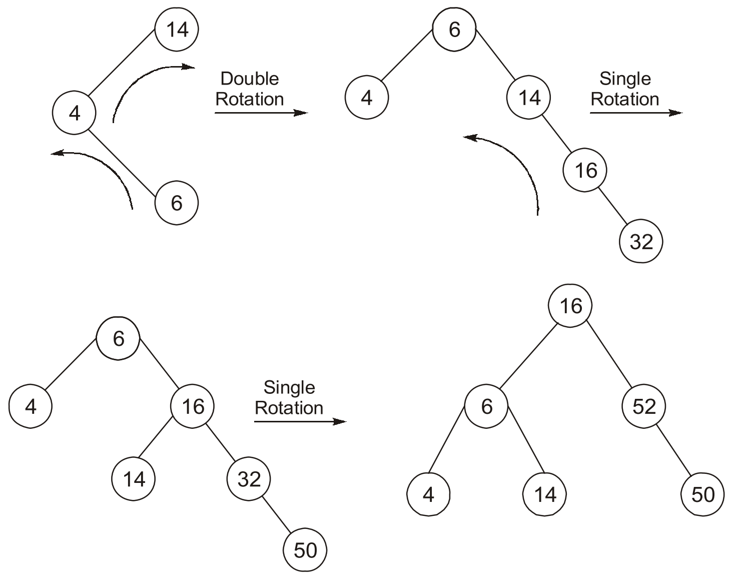
30. **(b)**

The program uses dynamic scoping. So if the variable is not available in the present function, then it search back in the previous function scope from where it was called.



Hence the output is 4, 6, 5 and 3 printed.

31. **(3)**

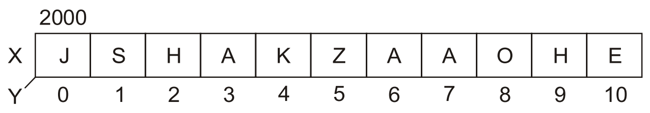


One double and two single rotations are required. So total 3 rotations.

32. **(183)**

$$\begin{aligned} \text{Loc}[-2][50] &= \text{Base address} + [(-2+5) * (50-5+1) + (50-5)] * 1 \\ &= 0 + 3 * 46 + 45 = 183 \end{aligned}$$

33. **(c)**



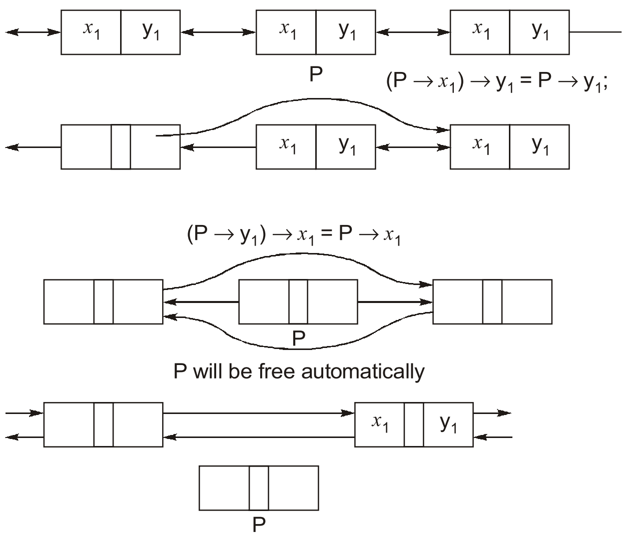
$$\begin{aligned} x &= 2000 \\ y &= 2000 \\ y[10] &= \text{E (69 in ASCII)} \\ y[7] &= \text{A (65 in ASCII)} \\ & \quad x + y[10] - y[7] \\ &= 2000 + 69 - 65 = 2004 \end{aligned}$$

Therefore it prints from the array starting at address; 2004 to the end i.e., “K Z A A O H E”.

34. **(b)**

This code returns the minimum of two integers *i* and *j*.

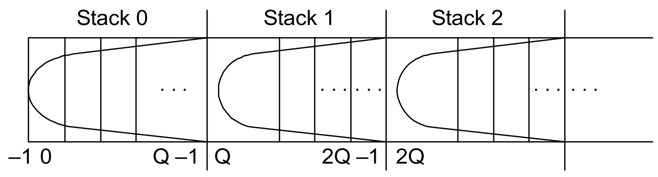
35. **(c)**


x_1
y_1
P
$(P \to x_1) \to y_1 = P \to y_1$;
$(P \to y_1) \to x_1 = P \to x_1$
P will be free automatically
P

36. **(d)**

While performing push operation on to stack we always perform "Overflow condition check", to ensure if stack is full or not.

Number of stacks possible is $\frac{P}{Q}$.



The initial value of stack pointer $T_i = \frac{P}{Q} * i - 1$

First element of next stack pointer $T_{i+1} = \frac{P}{Q}(i+1)$

$\therefore$ Overflow condition for stack i is $(T_i = \frac{P}{Q}(i+1) - 1)$

37. **(22)**

This function subtracts 1 from all nodes that containing odd values.
Sum of internal nodes will be

$$4 + 0 + 6 + 6 + 4 + 2 = 22$$

38. **(b)**

Consider Random array $a[\ ] = \{1, -2, 1, 1, -2, 1\}$
Output is 2 i.e. $\{1, 1\} = 2$
Consider Random array $a[\ ] = \{-2, -3, 4, -1, -2, 1, 5\}$
Output is 7 i.e. $\{4, -1, -2, 1, 5\} = 7$
i.e. sum of largest sum of contiguous sub array.

39. **(11)**

int z $\boxed{\cancel{0}\,\cancel{5}\,9}$

int count $\boxed{\cancel{0}\,\cancel{2}\,\cancel{4}\,\cancel{6}\,7}$ since static variable by default initialize to '0'.

1. $z = 3 + \text{value}\ (3)$

$\quad\hookrightarrow$ Count number of 1's in binary of 3 i.e. 2 (011)

$z = 3 + 2 = 5$

2. $z = 5 + \text{value}\ (5)$

$\quad\hookrightarrow$ Count number of 1's in binary of 5 i.e. 2 (101) + old value of count

$z = 5 + 4 = 9$

3. $z = 6 + \text{value}\ (6)$

$\quad\hookrightarrow$ Count number of 1's in binary of 6 i.e. 2 (110) + old value of count

$z = 6 + 6 = 12$

4. $z = 4 + \text{value}\ (4)$

$\quad\hookrightarrow$ Count number of 1's in binary of 4 i.e. 1 (100) + old value of count

$z = 4 + 7 = 11$

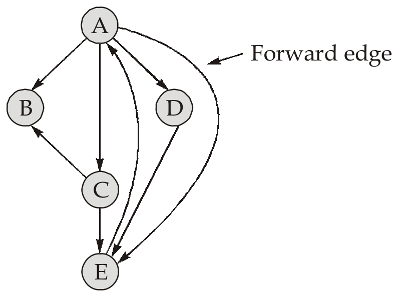
40. **(c)**

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.
$\therefore$ 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.
$\therefore$ A is path($x \to$ left, len – 1) || path($x \to$ right, len – 1)
If one of the path returns non-zero then it recursively returns back.
So option (c) is correct.

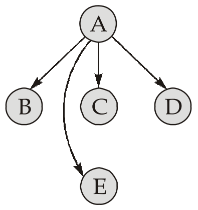
41. **(a)**

Since for undirected graph, breadth first search does not have back edge and forward edge but for directed graph we have back edge.

Ex: Consider Random Graph (Directed):



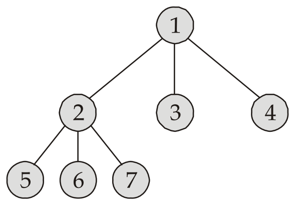
BFS of graph: Assume (A) is start node.



Here in graph no forward edge present, but E $\rightarrow$ A back edge present, so (b) is false.
Since undirected graph for BFS does not create back edge so statement is false.

42. **(b)**

Consider 3 array heap:



Array = A

1	2	3	4	5	6	7	....	
1	2	3	4	5	6	7	....	n

So, root at A[1] i.e. $d(i-1)+j+1$

$$= 3(1-1)+j+1$$
$$= 0+0+1 = 1$$

1st child of root $= d(i-1)+j+1$

$$= 3(1-1)+1+1 = 2$$

3rd child of root $= d(i-1)+j+1$

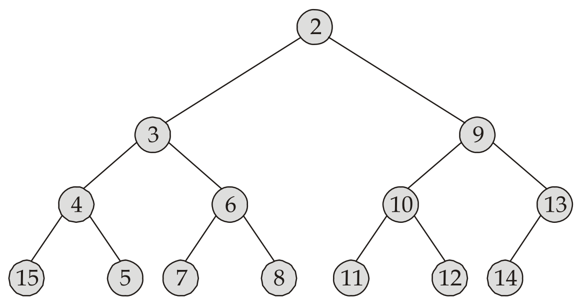
$$= 3(1-1)+3+1$$
$$= 0+4 = 4$$

Index for node 7 $= d(i-1)+j+1$

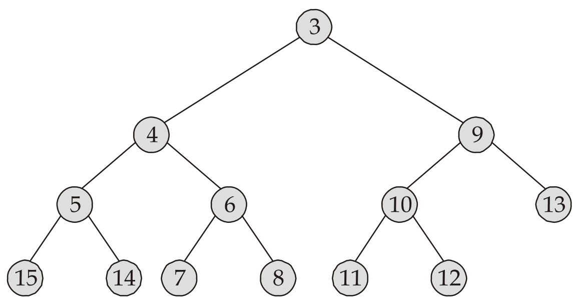
$$= 3(2-1)+3+1$$
$$= 3+3+1 = 7$$

So, index maps to $d(i-1)+j+1$.

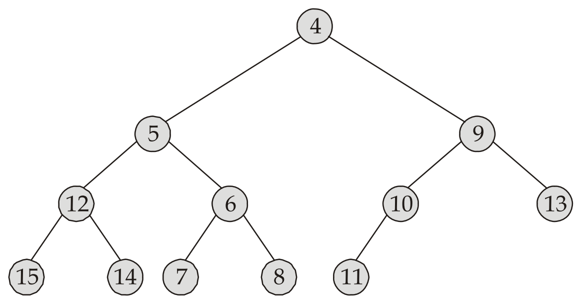
43. (18)



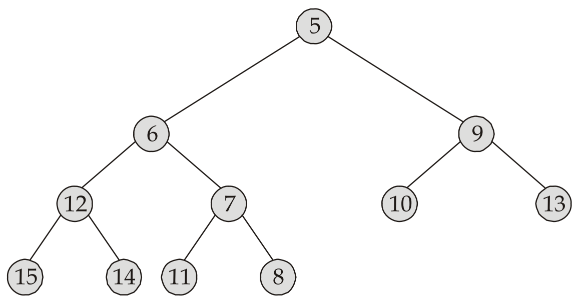
First deletion takes 4 node movement



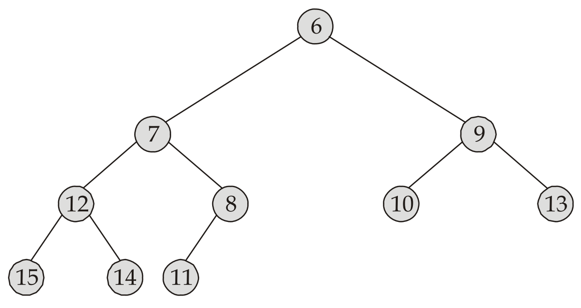
Second deletion takes 4 node movement



Third deletion takes 3 node movement



Fourth deletion takes 4 node movement



Fifth deletion takes 3 node movement

Total number of head movement = 4 + 4 + 3 + 4 + 3 = 18

44. (c)

Option (c) is the correct option, as in the first scenario, if the pointer is provided to the last node, then unless we have the starting address or address of second last node of the linked list, we cannot delete the last node. But any node can be deleted in the 2nd scenario.

45. (b)

The program X prints the ternary equivalent of 1023. Program P_1 also prints the ternary equivalent of 1023. However, program P_2 prints the ternary equivalent of 1023 in reverse order.

Hence the answer is (b).

46. **(b)**

Probability that last k slots are empty after first r iterations

$$= \frac{\overbrace{(N-k)(N-k)(N-k).....(N-k)}^{r \text{ times}}}{N^r}$$

$$= \frac{(N-k)^r}{N^r} = \left[\frac{N-k}{N}\right]^r = \left(1-\frac{k}{N}\right)^r$$

47. **(1365)**

The value of X i.e. number of stars printed can be represented by the following recurrence.

$$\text{Number of stars } (n) = \begin{cases} 1+4\cdot\text{Number of stars } (n/4); & n>1 \\ 1; & \text{otherwise} \end{cases}$$

Taking n as power of 4[$n = 4^k$]

Number of stars (4^k) = 1 + 4 number of stars (4^{k-1})

Solving the recurrence, we get

$$X = \left[\frac{4^{k+1}-1}{3}\right]$$

Now since $1024 = 2^{10} = 4^5$, put $k = 5$ in the above expression to get,

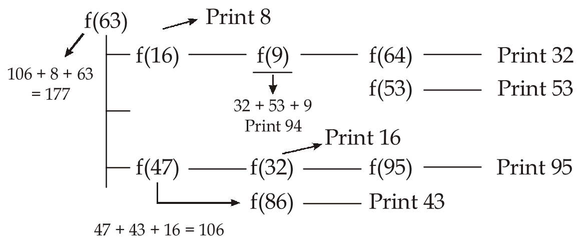
$$X = \frac{4^{5+1}-1}{3} = 1365$$

48. **(1252)**

$$\text{Loc}(A[100]\ [55]) = [\underset{\substack{\downarrow \\ \text{100 elements} \\ \text{in first row}}}{100} + \underset{\substack{\downarrow \\ \text{1 elements} \\ \text{in all rows} \\ \text{except 1st} \\ \text{and last}}}{(99-2+1)1} + (55-1) + \underset{\substack{\downarrow \\ \text{Base address}}}{1000}]$$

$$= 1252$$

49. **(256)**



Print output: 32, 53, 94, 8, 95, 16, 43, 106, 177
Sum of even values: 32 + 94 + 8 + 16 + 106 = 256

50. **(a, b, c)**

■■■■