7 Data Structures

1. Arrays

01. A one dimensional array A has indices 1..... 75. Each element is a string and takes up three memory words. The array is stored at location 1120 decimal. The starting address of A [49] is
(a) 1267 (b) 1164
(c) 1264 (d) 1169
[ISRO - 2009]

02. Let A(1:8, −5:5, −10:5) be a three dimensional array. How many elements are there in the array A?
(a) 1200 (b) 1408
(c) 33 (d) 1050
[ISRO - 2013]

03. In a compact one dimensional array representation for lower triangular matrix (all elements above diagonal are zero) of size n × n, none zero elements of each row are stored one after another starting from first row, the index of $(i, j)^{th}$ element in this new representation is
(a) i + j (b) $(j-1)+\frac{i(i-1)}{2}$
(c) i + j − 1 (d) $i+\frac{j(j-1)}{2}$
[ISRO - 2017(Dec)]

04. Consider a 2-dimensional array x with 10 rows and 4 columns, with each element storing a value equivalent to the product of row number and column number. The array is stored in row major format. If the first element x [0][0] occupies the memory location with address 1000 and each element occupies only one memory location, which all locations (in decimal) will be holding a value of 10?
(a) 1018, 1019 (b) 1022, 1041
(c) 1013, 1014 (d) 1000, 1399
[ISRO - 2020]

KEY & Detailed Solutions			
01. (d)	02. (b)	03. (b)	04. (b)

01. Ans:(d)
Sol: Start address of the element = base address of array + (Given index-starting index) * size of each element
A[49] = 1120 + (49-1) * 3 = 1264

02. Ans: (b)
Sol: No. of elements = upper bound-lower bound + 1
No. of elements in 1st dimension = upper limit - lower limit + 1 = 8 − 1 + 1 = 8
No. of elements in 2nd dimension
= 5 − (−5) + 1 = 11
No. of elements in 3rd dimension
= 15 − (− 10) + 1 = 16
Total No. of elements in the array
= 8 * 11 * 16 = 1408

03. Ans: (b)
Sol: For Row major order
Location of A[i, j] = Base Address +
{ (i–Lower index of row)* (i– Lower index of row +1)/2 + (j – Lower index of Col)} * C
We can modify the above formula for the index value in 1 D array
Index value= (i–Lower index of row)* (i– Lower index of row +1)/2 + (j– Lower index of Col)
(if lower bound is 0,0)
index = [i*(i +1)/2 + j]
(if lower bound is 1, 1)
index = [i*(i – 1)/2 + j – 1]

04. Ans: (b)
Sol: 0 Row [0..... 9] col[0..... 3]
each element is storing a value equivalent to the product of row number & col Number so 10 can be stored.

Array [5][2]

Array [2][5] → Not possible because Max col index is 3

Array [10][1] →

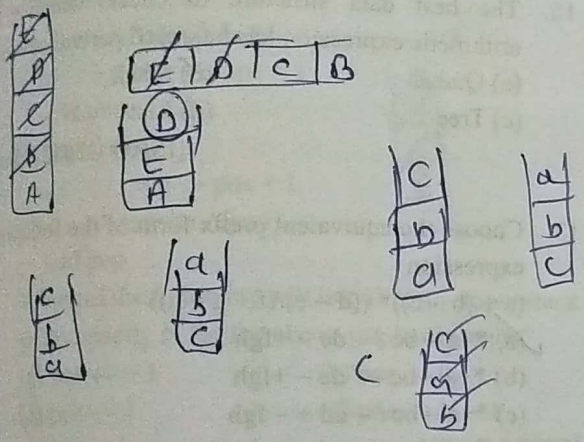
Now calculate Loc (Arr[5][2])

$= 1000 + 1[4*(5-0) + (2-0)]$

$= 1022$

Loc(Arr [10][1]) $= 1000 + 1[4*(10-0) + (1-0)]$

$= 1041$



A + (B − C) * D

A + − BC * D

A + • * − BCD

+ A • * − BCD

a + b * c − d ^ e ^ f

a + b * c − d ^ * e f ^

a + b * c − d e f ^ ^

a + bc * − d e f ^ ^

a b c * + d e f ^ ^ −

2. Stacks & Queues

01. The five items: A, B, C, D and E are pushed in a stack, one after the other starting from A. The stack is popped four times and each element is inserted in a queue. Then two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped item is

(a) A (b) B

(c) C (d) D

[ISRO - 2007]

02. Stack A has the entries a, b, c (with a on top). Stack B is empty. An entry popped out of stack A can be printed immediately or pushed to stack B. An entry popped out of the stack B can only be printed. In this arrangement, which of the following permutations of a, b, c are not possible?

(a) b a c (b) b c a

(c) c a b (d) a b c

[ISRO - 2008]

03. Assume that the operators +, −, x are left associative and ∧ is right associative. The order of precedence (from highest to lowest) is ∧, x, +, −. The postfix expression corresponding to the infix expression a + b x c − d ∧ e ∧ f is

(a) abc x + def ∧ ∧ − (b) abc x + de ∧ f ∧

(c) ab+c x d − e ∧ f ∧ (d) − + a x bc ∧ ∧ def

[ISRO - 2009]

04. The infix expression A + (B − C) * D is correctly represented in prefix notation as

(a) A + B − C * D (b) + A * −BCD

(c) ABC −D * + (d) A + BC − D*

[ISRO - 2009]

05. The five items: A, B, C, D, and E are pushed in a stack, one after the other starting from A. The stack is popped four times and each element is inserted in a queue. Then two elements are deleted from the

queue and pushed back on the stack. Now one item is popped from the stack. The popped item is
(a) A (b) B (c) C (d) D
[ISRO - 2009]

06. The expression 1*2^3*4^5*6 will be evaluated as
(a) 32^{30} (b) 162^{30} (c) 49152 (d) 173458
[ISRO - 2009]

07. Consider a standard circular queue 'q' implementation (which has same condition for queue full and queue empty) whose size is 11 and the elements of the queue are q[0], q[1],..... q[10]. The front and rear pointers are initialized to point at q[2]. In which position will the ninth element be added?
(a) q[0] (b) q[1] (c) q[9] (d) q[10]
[ISRO - 2014]

08. The five items: A, B, C, D, and E are pushed in a stack, one after the other starting from A. The stack is popped four times and each element is inserted in a queue. Then two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped item is _____.
(a) A (b) B (c) C (d) D
[ISRO - 2014]

09. If the sequence of operations - push (1), push (2), pop, push (1), push (2), pop, pop, pop, push (2), pop are performed on a stack, the sequence of popped out values
(a) 2, 2, 1, 1, 2 (b) 2, 2, 1, 2, 2
(c) 2, 1, 2, 2, 1 (d) 2, 1, 2, 2, 2
[ISRO - 2015]

10. The queue data structure is to be realized by using stack. The number of stacks needed would be
(a) It cannot be implemented
(b) 2 stacks
(c) 4 stacks (d) 1 stack
[ISRO - 2015]

11. The following postfix expression with single digit operands is evaluated using a stack
8 2 3 ^ / 2 3 * + 5 1 * –
(Note that ^ is the exponential operator)
The top two elements of the stack after the first * operator is evaluated are
(a) 6, 1 (b) 5, 7
(c) 3, 2 (d) 1, 5
[ISRO - 2016]

12. The best data structure to check whether an arithmetic expression has balanced parenthesis is a
(a) Queue (b) Stack
(c) Tree (d) List
[ISRO - 2017(May)]

13. Choose the equivalent prefix form of the following expression
(a + (b – c))* ((d – e)/(f + g ÷ h))
(a) *+a ÷ bc / – de – +fgh
(b) *+a – bc –/ de – +fgh
(c) *+a – bc / – ed + – fgh
(d) *+ab – c / – ed + – fgh
[ISRO - 2017(May)]

14. Which of the following permutation can be obtained in the same order using a stack assuming that input is the sequence 5, 6, 7, 8, 9 in that order?
(a) 7, 8, 9, 5, 6 (b) 5, 9, 6, 7, 8
(c) 7, 8, 9, 6, 5 (d) 9, 8, 7, 5, 6
[ISRO - 2017(Dec)]

15. The minimum number of stacks needed to implement a queue is
(a) 3 (b) 1 (c) 2 (d) 4
[ISRO - 2017(Dec)]

16. A language with string manipulation facilities uses the following operations.
head(s)- returns the first character of the string s
tail(s)- returns all but the first character of the strings

concat(s1, s2)- concatenates string s1 with s2.
The output of concat(head(s), head(tail(tail(s)))), where s is acbc is
(a) ab (b) ba (c) ac (d) aa

[ISRO - 2018]

17. A stack is implemented with an array of 'A [0..N-1]' and a variable 'pos'. The push and pop operations are defined by the following code.

```
push (x)
        A[pos] ← x
        pos ← pos – 1
end push
pop ()
        pop ← pos + 1
        return A[pos]
end pop
```

Which of the following will initialize an empty stack with capacity N for the above implementation?
(a) pos ← – 1 (b) pos ← 0
(c) pos ← 1 (d) pos ← N – 1

[ISRO - 2020]

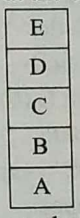
18. Convert the pre-fix expression to in-fix
–* +ABC*–DE+FG
(a) (A–B)*C+(D*E)–(F+G)
(b) (A+B)*C–(D–E)*(F–G)
(c) (A+B–C)*(D–E)*(F+G)
(d) (A+B)*C–(D*E)–(F+G)

[ISRO - 2020]

KEY & Detailed Solutions				
01. (d)	02. (c)	03. (a)	04. (b)	05. (d)
06. (c)	07. (a)	08. (d)	09. (a)	10. (b)
11. (a)	12. (b)	13. (a)	14. (c)	15. (c)
16. (a)	17. (d)	18. (*)		

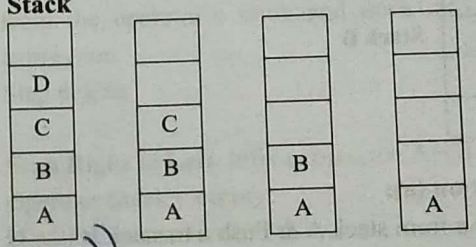
01. Ans: (d)

Sol: When five items: A, B, C, D and E are pushed in a stack: Order of stack becomes: A, B, C, D and E (A at the bottom and E at the top.)



stack is popped four items and each element is inserted in a queue: Order of queue: B, C, D, E (B at rear and E at the front) Order of stack after pop operations = A.

Stack



Queue

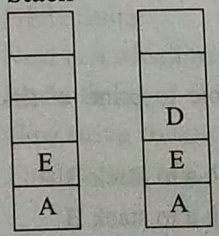
B	C	D	E

Two elements deleted from the queue and pushed back on the stack:
New order of stack = A, E, D (A at the bottom, D at the top).

Queue

		B	C

Stack



As D is on the top so when pop operation occurs D will be popped out. So, correct option is (d)

02. Ans:(c)
Sol:

Stack A

Stack B

Option (a):
Pop a from stack A & Push a to stack B
Pop b from stack A and Print b
Pop a from stack B and Print a
Pop c from stack A and Print c
Order = b a c

Option (b):
Pop a from stack A and Push a to stack B
Pop b from stack A and Print b
Pop c from stack A and Print c
Pop a from Stack B and Print a
Order = b c a

Option (c):
this sequence is not possible to print 'c' first we need to
Pop a from stack A and Push a to stack B
Pop b from stack A and Push a to stack B
Pop c from stack A and print c
Pop b from Stack B and print B
So after c only b can be printed but option c is printing a after c which is not possible

Option (d)
Pop a from stack A and Print a
Pop b from stack A and Print b
Pop c from stack A and Print c

03. Ans: (a)
Sol: Algorithm

- **Step 1:** Scan the Infix Expression from left to right.
- **Step 2:** If the scanned character is an operand, append it with final Infix to Postfix string.
- **Step 3:** Else,
- **Step 3.1:** If the precedence order of the scanned(incoming) operator is greater than the precedence order of the operator in the stack (or the stack is empty or the stack contains a '(' or '[' or '{'), push it on stack.
- **Step 3.2:** Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
- **Step 4:** If the scanned character is an '(' or '[' or '{', push it to the stack.
- **Step 5:** If the scanned character is an ')'or ']' or '}', pop the stack and and output it until a '(' or '[' or '{' respectively is encountered, and discard both the parenthesis.
- **Step 6:** Repeat steps 2-6 until infix expression is scanned.
- **Step 7:** Print the output
- **Step 8:** Pop and output from the stack until it is not empty.

Here, the infix expression is
a + b × c − d ^ e ^ fa + b × c − d ^ e ^ f
a: it is operand so print it
+: stack is empty so push into the Operator Stack
b: it is operand so print it
*: its having higher precedence than + so push into Operator Stack

c : it is operand so print it
−: is having less precedence than '*' so pop from operator stack and print *. after this stack will be having + on top. which is having same precedence as − but both are left to right associative so pop + and print it. Now stack is empty so we can Push − to it.
d: it is operand so print it
^: top of the stack is having − and ^ has higher precedence than −. So simply push ^ into operator stack
e: it is operand so print it.
^: Now top of the stack is also ^. Operator ^ is right associative so ^ will be pushed.
f : it is operand so print it.
Now, we have scanned entire infix expression. Now pop the stack until it becomes empty.This way you will get abc*+def ^ ^−

04. **Ans: (b)**

Sol: Algorithm

Iterate the given expression from left to right, one character at a time

Step 1: First reverse the given expression

Step 2: If the scanned character is an operand, put it into prefix expression.

Step 3: If the scanned character is an operator and operator's stack is empty, push operator into operators' stack.

Step 4: If the operator's stack is not empty, there may be following possibilities.

(a) If the precedence of scanned operator is greater than the top most operator of operator's stack, push this operator into operator 's stack.

(b) If the precedence of scanned operator is less than the top most operator of operator's stack, pop the operators from operator's stack untill we find a low precedence operator than the scanned character.

(c) If the precedence of scanned operator is equal then check the associativity of the operator. If associativity left to right then simply put into stack. If associativity right to left then pop the operators from stack until we find a low precedence operator.

(d) If the scanned character is opening round bracket ('('), push it into operator's stack.

(e) If the scanned character is closing round bracket (')'), pop out operators from operator's stack until we find an opening bracket ('(').

Repeat Step 2, 3 and 4 till expression has character

Step 5: Now pop out all the remaining operators from the operator's stack and push into postfix expression.

Step 6: Exit

Scan Right to Left infix expression A+(B − C)* D
Operator Stack : empty
D will be printed
Prefix String: D
* Will be pushed in stack
) will be pushed in operator stack
C will be printed
Prefix String: CD
– will be pushed in operator stack
B will be printed so
Prefix String: BCD
Encountered "(" so Pop operator stack upto ")"
Prefix String: –BCD
Next is + has lower priority than * so pop *
Prefix String: *–BCD
Next is A which will be printed so
Prefix String: +A*–BCD
Now string complete so pop all element of stack
Final Output : +A*–BCD

05. **Ans: (d)**

Sol: When five items: A, B, C, D, and E are pushed in a stack: Order of stack becomes: A, B, C, D and E (A at the bottom and E at the top.)

E
D
C
B
A

stack is popped four items and each element is inserted in a queue: Order of queue: B, C, D, E (B at rear and E at the front) Order of stack after pop operations = A.

Stack

D			
C	C		
B	B	B	
A	A	A	A

Queue

B	C	D	E

Two elements deleted from the queue and pushed back on the stack:

New order of stack = A, E, D (A at the bottom, D at the top).

Queue

		B	C

Stack

	D
E	E
A	A

As D is on the top so when pop operation occurs D will be popped out. So, correct option is (D)

E
A

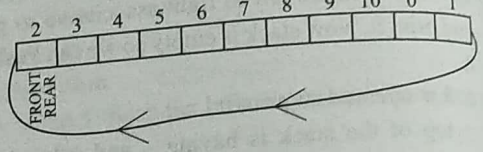
06. Ans: (c)

Sol: = 1 * (2^3) * (4^5) * 6 because ^ has higher priority than *

=1 * 8 * 1024 * 6 all operator has same priority but left to right associativity = 49152

07. Ans: (a)

Sol: Circular queue whose total size is 11, front and rear pointers are initialized to point at q[2]:



For enqueue, we increment the REAR pointer and then insert an element.

So 1st element will be inserted at q[3], 2nd element will be inserted at q[4]..... 9th element will be inserted at q[0].

08. Ans:(d)

Sol: When five items: A, B, C, D, and E are pushed in a stack: Order of stack becomes: A, B, C, D and E (A at the bottom and E at the top.)

E
D
C
B
A

stack is popped four items and each element is inserted in a queue: Order of queue: B, C, D, E (B at rear and E at the front) Order of stack after pop operations = A.

Stack

D			
C	C		
B	B	B	
A	A	A	A

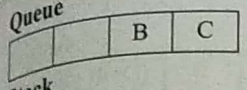
Queue

B	C	D	E

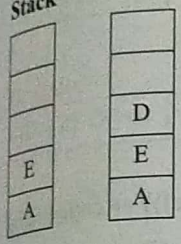
Two elements deleted from the queue and pushed back on the stack:

New order of stack = A, E, D (A at the bottom, D at the top).

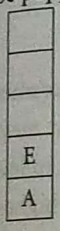
Queue



Stack



As D is on the top so when pop operation occurs D will be popped out. So, correct option is (D)



09. Ans: (a)

Sol: Right hand side of the row is representing Top of the stack.

Operation	Stack	Pop Sequence
Push 1	1	
Push 2	1,2	
Pop	1	2
Push 1	1, 1	
Push 2	1, 1, 2	
Pop	1,1	2,2
Pop	1	2, 2, 1
Pop	Empty	2, 2, 1, 1
Push 2	2	
Pop	Empty	2, 2, 1, 1, 2

10. Ans: (b)

Sol: A Queue is defined by its property of FIFO, which means First in First Out. For performing enqueue we require only one stack as we can directly push data onto the stack but if we pop stack we will get elements in LIFO. To solve this problem, we will take one more stack.

To perform dequeue, we pop elements from stack 2. If stack 2 is empty then first we push all the elements of stack 1 and push them to empty stack2 then we perform pop from stack 2.

11. Ans: (a)

Sol: Postfix evaluation can be implemented with the help of operand stack, and by the use of following steps.

1. Read all the symbols one by one from left to right in the given Postfix Expression.
2. If symbol is operand, then push it on to the Stack.
3. If the symbol is operator then perform TWO pop operations.
   Operand2 = Pop(operand stack)
   Operand1 = Pop(operand stack)
   Result = (Operand1) operator (Operand2).
   Push Result into operator stack.
4. Repeat above steps till the end of the expression.

INPUT	OPERATION	STACK	CALCULATION
8	Push	8	
2	Push	8,2	
3	Push	8,2,3	
^	Pop & evaluate	8	$2 \wedge 3 = 8$
	Push result	8,8	
/	Pop & evaluate	empty	8/8 = 1
	Push result	1	
2	Push	1, 2	
3	Push	1, 2, 3	
*	Pop & evaluate	1	2*3 = 6
	Push result	1,6	

After evaultion of * operator we push result 6 inside the stack. Now top of the operand stack is 6 then 1. So answer is (A).

12. Ans: (b)

Balanced Parenthesis is one of the application of Stack.

13. Ans: (a)

Sol: (a+(b−c))*((d−e)/(f+g−h))
(a+(b−c))*((−de)/((+fg)−h))
(a+(−bc))*((−de)/(−+fgh))
(+a−bc)*(/−de−+fgh)
*+a−bc/−de−+fgh

14. Ans: (c)

Sol: We need to try all the options one by one.
Lets try to obtain the sequence in option (a)
Push 5,6,7, Pop 7, Push 8, Pop 8, Push 9, Pop 9, now if we implement Pop then we will get 6 so option a is not possible.
Lets try to obtain the sequence in option (b)
Push 5, Pop 5 Push 6, 7, 8, 9 Pop 9, now if we implement Pop then we will get 8 so option b is not possible.
We can obtain the sequence by performing operations in the manner:
Lets try to obtain the sequence in option (c)
Push 5, 6, 7, Pop 7, Push 8, Pop 8, Push 9, Pop 9, Pop 6, Pop 5.
hence the sequence will be 7, 8, 9, 6, 5.
Lets try to obtain the sequence in option (d)
Push 5, 6, 7, 8, 9 Pop three times we will get 9, 8 7 now if we implement Pop then we will get 6 so option d is not possible.
The sequence given in option (c) is one of the only possible sequences which can be obtained.

15. Ans: (c)

Sol: A queue can be implemented using two stacks. Queue can be implemented with 2 stacks in two ways:

Method 1 (By making enQueue operation costly)
This method makes sure that newly entered element is always at the top of s1, so that deQueue operation just pops from stack1. To put the element at top of stack1, stack2 is used.

Method 2 (By making deQueue operation costly)
In this method, in en-queue operation, the new element is entered at the top of stack1. In de-queue operation, if stack2 is empty then all the elements are moved to stack2 and finally top of stack2 is returned.

16. Ans:(a)

Sol: s = acbc
head(s) = a
tail(s) = cbc
tail(tail(s)) = bc
head(tail(tail(s))) = b
concat(head(s),head(tail(cbc))) → concat(a,b)
ab

17. Ans: (d)

Sol: pos←N-1
In given code, push operation decrement pos and Pop operation increment pos. That means stack is traversing array from higher index to lower index. So initially pos should be at the highest index of array pos←N-1.

18. Ans: None of the option is correct

Sol: Let's evaluate the prefix
–* + ABC* – DE + FG
Traverse Right to left
OPERAND Stack
Insert F G

Top

F	G					

Evaluate +

(F+G)						

Insert E D

Top

(F+G)	E	D				

Evaluate –

Top

(F+G)	(D –E)					

Evaluate*

(D – E)* (F+G)						

Insert C B A

(D – E)* (F+G)	C	B	A		

Evaluate +

(D – E)* (F+G)	C	(A+B)			

Evaluate*

(D – E)* (F+G)	(A+B)* C				

Evaluate –

((A+B)*C) – ((D-E)* (F+G))

(A+B)*C – (D–E)* (F+G)

3. Linked List

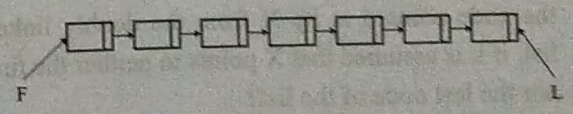
01. The time required to search an element in a linked list of length n is
(a) $O(\log_2 n)$ (b) O (n)
(c) O(1) (d) $O(n^2)$ **[ISRO - 2008]**

02. Which of the following operations is performed more efficiently by doubly linked list than by linear linked list?
(a) Deleting a node whose locations is given
(b) Searching an unsorted list for a given item
(c) Inserting a node after the node with a given location
(d) Traversing the list to process each node.
[ISRO - 2008]

03. The minimum number of fields with each node of doubly linked list is
(a) 1 (b) 2 (c) 3 (d) 4
[ISRO - 2008]

04. The following steps in a linked list
p = getnode()
info (p) = 10
next (p) = list
list = p
result in which type of operation?
(a) pop operation in stack
(b) removal of a node
(c) inserting a node
(d) modifying an existing node **[ISRO - 2013]**

05. Consider a single linked list where F and L are pointers to the first and last elements respectively of the linked list. The time for performing which of the given operations depends on the length of the linked list?



(a) Delete the first element of the list
(b) Interchange the first two elements of the list
(c) Delete the last element of the list
(d) Add an element at the end of the list.

[ISRO - 2014]

06. Given two statements
(i) Insertion of an element should be done at the last node in a circular list
(ii) Deletion of an element should be done at the last node of the circular list
(a) Both are true
(b) Both are false
(c) First is false and second is true
(d) None of the above

[ISRO - 2017(May)]

07. Which of the following data structure is useful in traversing a given graph by breadth first search?
(a) Stack (b) Queue
(c) List (d) None of the above

[ISRO - 2017(May)]

08. In a doubly linked list, the number of pointers affected for an insertion operation will be
(a) 4
(b) 0
(c) 1
(d) Depends upon the nodes of the doubly linked list

[ISRO - 2017(May)]

09. A doubly linked list is declared as

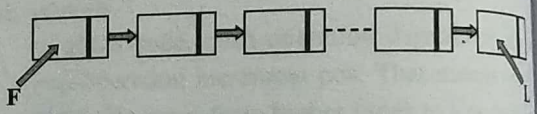
```
struct Node {
        int Value ;
        struct Node *Fwd ;
        struct Node *Bwd ;
} ;
```

Where **Fwd** and **Bwd** represent forward and backward link to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by **X** from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

(a) X→Bwd→Fwd = X→Fwd; X→Fwd→Bwd = X→Bwd;
(b) X→Bwd.Fwd = X→Fwd ; X.Fwd→Bwd = X→Bwd;
(c) X.Bwd→Fwd = X.Bwd ; X→Fwd.Bwd = X.Bwd;
(d) X→Bwd→Fwd = X→Bwd ; X→Fwd→Bwd = X→Fwd;

[ISRO - 20

10. Consider a singly linked list of the form where F is a pointer to the first element in the linked list and L is the pointer to the last element in the list. The time of which of the following operations depends on the length of the list?



(a) Delete the last element of the list
(b) Delete the first element of the list
(c) Add an element after the last element of the list
(d) Interchange the first two elements of the list

[ISRO - 201

KEY & Detailed Solutions				
01. (b)	02. (a)	03. (c)	04. (c)	05. (c)
06. (b)	07. (b)	08. (*)	09. (a)	10. (a)

01. Ans: (b)

Sol: In Linked-list we can apply Linear search which takes O(n). In the worst case, the element to be searched has to be compared with all elements of linked list. It will take O(n) time to search the element.

02. Ans: (a)

Sol: Option (a), deleting a node whose location is given. Before we remove node x from the LL we need to store the address of next node in the next part of

previous node. Since we need previous node during the delete operation, doubly linked list increases the efficiency by decreasing the time required for the operation.

Option (b) and (d) cannot be true because in both the cases we need to visit every node one after other.

Option (c), to insert a new node in the LL after some specific node, we do not need anything with the previous node thus, doubly LL will serve no purpose in this case too. Instead we modify extra node to maintain previous node.

03. Ans: (c)

Sol: Each node of doubly link list always has 3 fields → the previous node pointer, the data field, and the next node pointer.

04. Ans:(c)

Sol: List is representing address of first node of the linked list.

P = get node (); mean dynamically allocating a space for a new node and returns a pointer.

info(p) = 10 mean value of the new node is equal to 10

next (p) = list means next part of new node is containing address of Linked List

list = p Updating starting address of the Linked list

So with given code we are inserting new node at the beginning of the linked list.

05. Ans:(c)

Sol: (a) Deleting the first element of the list will not depend on the length of the link list as save the first node as temp = F then Pointer F will be set to the next node by F = F→next and delete temp.

(b) Interchange the first two elements of the list does not require the length of Linked List.

Temp = F→data

F→data=F → next → data

F → next → data=temp

(c) Deleting the last element of the list will require traversal upto the length of the linked list to obtain the address of the 2nd last node.

(d) Add an element at the end of the list.

new node → next = null

L→ next = newnode;

06. Ans: (b)

Sol: Insertion and deletion can be done at the beginning, at the end or at the middle of the circular linked list.

07. Ans: (b)

Sol: Breadth-first search uses a Queue data structure. Depth-first search uses a Stack data structure.

08. Ans: should be None of these

Sol: In a doubly linked list the number of pointers affected for an insertion operation will depend upon the where we are inserting the node. There are 4 cases: insert at begin, insert at the end, insert at the middle. we have solved insert at begin and insert at the end to solve the above question

Case 1:

insertion at the beginning will affect 4 pointers

newnode → next = head;

newnode → prev = null;

head → prev = newnode;

head = newnode;

total change 4

Case 2: insertion at the end will affect 3 pointers

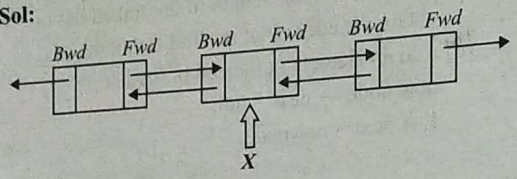
newnode → next = null;

newnode → prev = Lastnode;

lastnode → next = newnode

09. Ans: (a)

Sol:



We can remove X

$X \rightarrow Bwd \rightarrow Fwd = X \rightarrow Fwd$

$X \rightarrow Fwd \rightarrow Bwd = X \rightarrow Bwd$

Option (a) is correct

Option (b) and (c) are incoreect as they are using . operator

Option (d)

$X \rightarrow Bwd \rightarrow Fwd = X \rightarrow Bwd$; It will store the address of the previous node of x in the fwd part of previous node. So it makes the previous node to point itself.

$X \rightarrow Fwd \rightarrow Bwd = X \rightarrow Fwd$; It will store the address of next node of X in the Bwd part of the next node.

10. Ans: (a)

Sol:

(a) Deleting the last element of the list will require traversal up to the length of the linked list to obtain the address of the 2nd last node.

(b) Deleting the first element of the list will not depend on the length of the link list as save the first node as temp = F then Pointer F will be set to the next node by F = F→next and delete temp.

(c) Add an element at the end of the list.
L→next = newnode;
L = newnode;

(d) Interchange the first two elements of the list does not require the length of Linked List.
Temp = F→data
F→data = F → next → data
F → next → data=temp

4. Trees

01. A complete binary tree with the property that the value at each node is at least as large as the values at its children is known as
(a) binary search tree (b) AVL tree
(c) Completely balanced tree (d) Heap
[ISRO - 2008]

02. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?
(a) 7 5 1 0 3 2 4 6 8 9 (b) 0 2 4 3 1 6 5 9 8 7
(c) 0 1 2 3 4 5 6 7 8 9 (d) 9 8 6 4 2 3 0 1 5 7
[ISRO - 2009]

03. A data structure is required for storing a set of integers such that each of the following operations can be done in (log n) time, where n is the number of elements in the set.
1. Deletion of the smallest element.
2. Insertion of an element if it is not already present in the set.

Which of the following data structures can be used for this purpose?
(a) A heap can be used but not a balanced binary search tree
(b) A balanced binary search tree can be used but not a heap
(c) Both balanced binary search tree and heap can be used
(d) Neither balanced binary search tree nor heap can be used
[ISRO - 2009]

04. The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum distance of a leaf node from the root)?
(a) 2 (b) 3 (c) 4 (d) 6
[ISRO - 2009]

05. A full binary tree with n leaves contains
(a) n nodes (b) $\log_2$ n nodes
(c) 2n–1 (d) 2^n nodes
[ISRO - 2009]

06. How many distinct binary search trees can be created out of 4 distinct keys?
(a) 5 (b) 14
(c) 24 (d) 35
[ISRO - 2011]

07. If node A has three siblings and B is parent of A, what is the degree of A?
(a) 0 (b) 3 (c) 4 (d) 5
[ISRO - 2011]

08. The in-order traversal of a tree resulted in FBGADCE. Then the pre-order traversal of that tree would result in
(a) FGBDECA (b) ABFGCDE
(c) BFGCDEA (d) AFGBDEC
[ISRO - 2011]

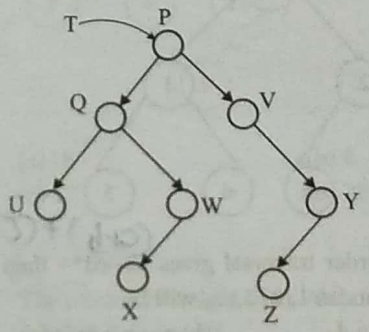
09. The average depth of a binary search tree is
(a) $O(n^{0.5})$ (b) O(n)
(c) O(log n) (d) O(n log n)
[ISRO - 2011]

10. Number of comparisons required for an unsuccessful search of an element in a sequential search organized, fixed length, symbol table of length L is
(a) L (b) L/2
(c) (L+1)/2 (d) 2L
[ISRO - 2011]

11. The number of rotations required to insert a sequence of elements 9, 6, 5, 8, 7, 10 into an empty AVL tree is?
(a) 0 (b) 1 (c) 2 (d) 3
[ISRO - 2013]

12. Which of the following number of nodes can form a full binary tree?
(a) 8 (b) 15 (c) 14 (d) 13
[ISRO - 2013]

13. Consider the following binary search tree T given below. Which node contains the fourth smallest element in T?



(a) Q (b) V (c) W (d) X
[ISRO - 2014]

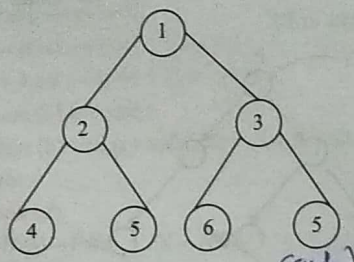
14. How many different tress are there with four nodes A, B, C and D?
(a) 30 (b) 60
(c) 90 (d) 120
[ISRO - 2014]

15. The number of spanning trees for a complete graph with seven vertices is
(a) 2^5 (b) 7^5 (c) 3^5 (d) $2^{2\times5}$
[ISRO - 2015]

16. Average number of comparison required for a successful search for sequential search on 'n' items is
(a) $\frac{n}{2}$ (b) $\frac{(n-1)}{2}$
(c) $\frac{(n+1)}{2}$ (d) None of the above
[ISRO - 2016]

17. A complete binary tree with n non-leaf nodes contains
(a) $\log_2$ n nodes (b) n+1 nodes
(c) 2n nodes (d) 2n+1 nodes
[ISRO - 2016]

18. Consider the following tree



If the post order traversal gives ab–cd*+ then the label of the nodes 1, 2, 3,..... will be
(a) +,–,*,a,b,c,d (b) a,–,b,+,c,–,d
(c) a,b,c,d,–,*,+ (d) –,a,b,+,*,c,d
[ISRO - 2017(May)]

19. The number of structurally different possible binary trees with 4 nodes is
(a) 14 (b) 12 (c) 336 (d) 168
[ISRO - 2017(Dec)]

20. Match the following and choose the correct answer in the order A, B, C

A. Heap Construction	p. O(n logn)
B. Hash table construction with linear probing	q. $O(n^2)$
C. AVL Tree construction	r. O(n)

(Bounds given may or may not be asymptotically tight)
(a) q, r, p (b) p, q, r
(c) q, p, r (d) r, q, p
[ISRO - 2017(Dec)]

21. A binary search tree is used to locate the number 43. Which one of the following probe sequence is not possible?
(a) 61, 52, 14, 17, 40, 43
(b) 10, 65, 31, 48, 37, 43
(c) 81, 61, 52, 14, 41, 43
(d) 17, 77, 27, 66, 18, 43
[ISRO - 2017(Dec)]

22. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the reversal ordering on natural numbers i.e. 9 is assumed to be smallest and 0 is assumed to be largest. The in-order traversal of the resultant binary search tree is
(a) 9, 8, 6, 4, 2, 3, 0, 1, 5, 7
(b) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
(c) 0, 2, 4, 3, 1, 6, 5, 9, 8, 7
(d) 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
[ISRO - 2017(Dec)]

23. A priority queue is implemented as a Max-heap. Initially it has 5 elements. The level order traversal of the heap is 10, 8, 5, 3, 2. Two new elements '1' and '7' are inserted into the heap in that order. The level order traversal of the heap after the insertion of the elements is
(a) 10, 8, 7, 5, 3, 2, 1
(b) 10, 8, 7, 2, 3, 1, 5
(c) 10, 8, 7, 1, 2, 3, 5
(d) 10, 8, 7, 3, 2, 1, 5
[ISRO - 2017(Dec)]

24. A strictly binary tree with 10 leaves
(a) cannot have more than 19 nodes
(b) has exactly 19 nodes
(c) has exactly 17 nodes
(d) has exactly 20 nodes
[ISRO - 2017(Dec)]

25. What is the maximum height of any AVL tree with 7 nodes? Assume that height of tree with single node is 0.
(a) 2 (b) 3 (c) 4 (d) 5
[ISRO - 2017(Dec)]

26. Which one of the following property is correct for a red-black tree?
(a) Every simple path from a node to a descendant leaf contains the same number of black nodes
(b) If a node is red, then one children is red and another is black
(c) If a node is red, then both its children are red
(d) Every leaf node (sentinel node) is red

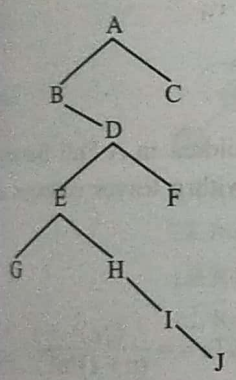
[ISRO - 2017(Dec)]

27. The in-order and pre-order traversal of a binary tree are d b e a f c g and a b d e c f g respectively. The post order traversal of a binary tree is
(a) e d b g f c a (b) e d b f g c a
(c) d e b f g c a (d) d e f g b c a

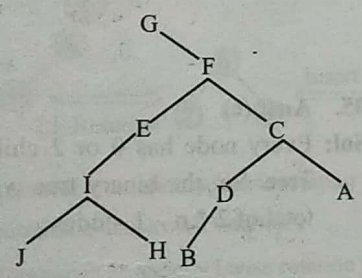
[ISRO - 2017(Dec)]

28. If Tree-1 and Tree-2 are the trees indicated below :

Tree - 1



Tree - 2



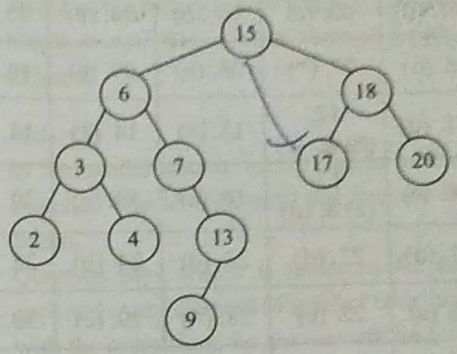
Which traversals of Tree-1 and Tree-2, respectively, will produce the same sequence?
(a) Preorder, Postorder (b) Postorder, inorder
(c) Postorder, Preorder (d) Inorder, preorder

[ISRO - 2018]

29. Given a binary-max heap. The elements are stored in an arrays as 25, 14, 16, 13, 10, 8, 12.. What is the content of the array after two delete operations?
(a) 14, 13, 8, 12, 10 (b) 14, 12, 13, 10, 8
(c) 14, 13, 12, 8, 10 (d) 14, 13, 12, 10, 8

[ISRO - 2018]

30. What is the in-order successor of 15 in the given binary search tree?



(a) 18 (b) 6
(c) 17 (d) 20

[ISRO - 2020]

31. The minimum height of an AVL tree with n nodes is
(a) Ceil ($\log_2(n+1)$)
(b) $1.44\log_2 n$
(c) Floor ($\log_2(n+1)$)
(d) $1.64\log_2 n$

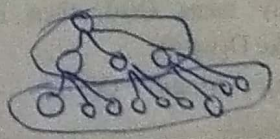
[ISRO - 2020]

32. The post-order traversal of a binary tree is ACEDBHIGF. The pre-order traversal is
(a) ABCDEFGHI
(b) FBADCEGIH
(c) FABCDEGHI
(d) ABDCEFGIH

[ISRO - 2020]

33. Of the following, which best approximates the ratio of the number of nonterminal nodes in the total number of nodes in a complete K-ary tree of depth N?
(a) 1/N (b) N–1/K
(c) 1/K (d) K –1/K

[ISRO - 2020]



KEY & Detailed Solutions				
01. (d)	02. (c)	03. (b)	04. (b)	05. (c)
6. (b)	07. (*)	08. (b)	09. (c)	10. (a)
11. (d)	12. (b) & (d)	13. (c)	14. (*)	15. (b)
16. (c)	17. (c) & (d)	18. (a)	19. (a)	20. (d)
21. (d)	22. (d)	23. (d)	24. (b)	25. (b)
26. (a)	27. (c)	28. (*)	29. (c)	30. (c)
31. (c)	32. (*)	33. (c)		

01. Ans: (d)

Sol: Heap which is a complete binary tree with every node has value more than or equal to its children in case of Max heap and every node has value less than or equal to its children in case of Min heap.

Option (a) BST is not needed to be CBT as it could be skewed as well.

Option (b) AVL tree is balanced BST but it need not to be Completed BST as it could increase at the right hand side before left hand side.

Option (c) complete binary trees are Completely Balanced tree but vice versa is not true.

02. Ans: (c)

Sol: The inorder sequence of a binary search tree always returns the numbers arranged in **ascending order**.

So, option (c) is correct

03. Ans: (b)

Sol: Heap is almost complete Binary tree.

Deletion of smallest element from Min Heap will take O(logn)time (1st removal of root element and replace with last element then balancing)

Finding a element is present/not and insertion: Finding only takes O(n), then insertion and balancing take O(logn).

So, total O(n) + O(logn) = O(n).

So heap cannot be used as Insertion of an element if it is not already present in the set is taking O(n) instead of O(logn)

Balanced search tree have height logn

Deletion of smallest element will take O(logn) time

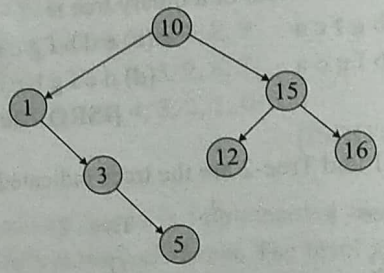
Finding a element is present/not and doing insertion: O(logn)

Balanced BST can serve the purpose.

so answer is (b) balanced binary search tree can be used but not a heap

04. Ans: (b)

Sol:



05. Ans: (c)

Sol: Every node has 0 or 2 children in A Full Binary Tree. So, the binary tree with n leaves contains a total of 2 * n – 1 nodes.

06. Ans: (b)

Sol: The number of different BSTs $= = \frac{(2n)!}{(n+1)!n!}$

Here n = 4,

Number of distinct BST's = $(4 \times 2)! / (4! \times 5!) = 14$

So, correct option is (B)

07. Ans: information is not sufficient to tell the degree of the A

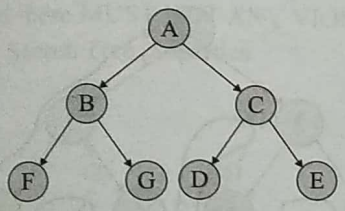
Sol: A has three siblings and B is parent of A, but no information is given about the children of A or if it is an internal node or a leaf node and also total number of node in the graph is not given. So The PROVIDED information is not sufficient to tell the degree of the A.

08. Ans: (b)

Sol: To find out preorder in binary tree, we need inorder and postorder traversal as input because, It is not possible to construct a binary tree with in-order alone. But in case of Binary search tree, it is possible.

Conceptually it is not possible to find out pre-order but we can try to create binary tree for given in-order with all given pre-order options one by one. Only combination of option B with given in-order is feasible.

If we try to generate tree from given In-order in FBGADCE and pre-order ABFGCDE (option b)



09. Ans: (c)

Sol: If there are n nodes in a binary search tree, maximum height of the binary search tree is n-1 and minimum height is ceil (log n) and average height is also O(log n)

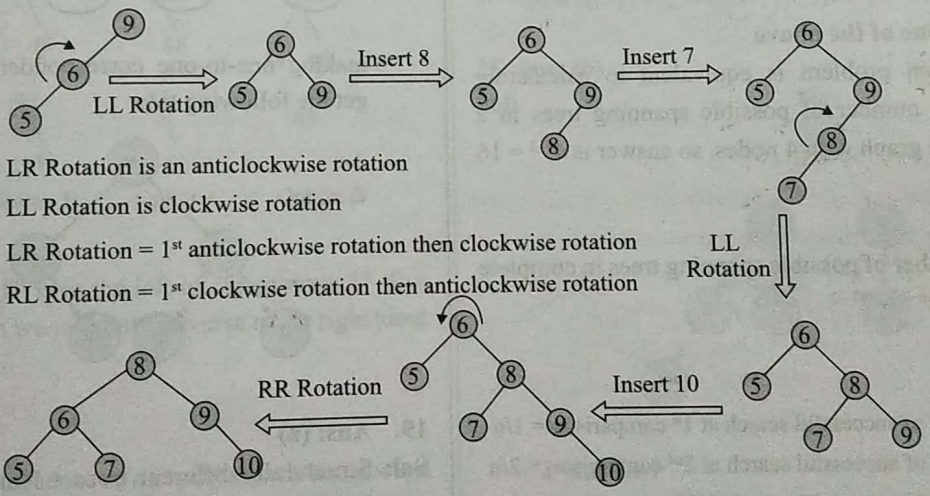
10. Ans: (a)

Sol: In Sequential search, each element of the table is searched one by one until the desired element is found.

Question says search is unsuccessful that means each and every element of symbol table is compared with the search key. So answer will be L.

11. Ans: (d)

Sol: The insertion and rotation of the various elements are shown in the following figure:



LR Rotation is an anticlockwise rotation

LL Rotation is clockwise rotation

LR Rotation = 1st anticlockwise rotation then clockwise rotation

RL Rotation = 1st clockwise rotation then anticlockwise rotation

12. Ans: (b) & (d)

Sol: A full binary tree can be defined as a binary tree in which all the nodes have 0 or two children i.e. a binary tree in which all the nodes have two children except the leaf nodes.

if I be internal nodes in a tree and L to be a leaf node in a tree, then the number of leaf nodes would be equal to:

$L = I + 1$

If Tree has, I number of internal nodes and N to be the total number of nodes, then the total number of nodes would be equal to:

$N = 2I + 1$

Both option b and d satisfied the condition of Full binary tree

13. Ans: (c)

Sol: Inorder traversal can be used to prints data in the sorted order but only if a given tree is a binary search tree.

inorder traversal:

- Recursively traverse the left subtree
- Visit the root node
- Recursively traverse the right subtree

```
void Inorder(struct node* root)
{
  if (root == NULL)
  return;
Inorder(root→left);
printf(" %d ", root→data);
Inorder(root→right);
}
```

inorder traversal → UQXWPVZY

So 4th smallest element is W.

14. Ans: None of the above

Sol: The given problem is equivalent to calculating the total number of possible spanning trees in a complete graph with 4 nodes, so answer is $4^{4-2} = 16$

15. Ans: (b)

Sol: Total number of possible spanning trees in complete graph is $n^{n-2} = 7^{7-2}$

16. Ans: (c)

Sol: Probability of successful search at 1st comparison = 1/n

Probability of successful search at 2nd comparison = 2/n

Probability of successful search at 3rd comparison = 3/n

Expected number of comparisons for successful search

$= 1/n + 2/n + 3/n + \ldots\ldots\ldots + n/n$

$= (1+2+3+\ldots\ldots+n) / n$

$= (n*(n+1)/2) / n$

$= (n+1)/2$

17. Ans: (c) & (d)

Sol: In a complete binary tree, each level is filled from left to right and all levels except the last level are fully filled.

In complete binary

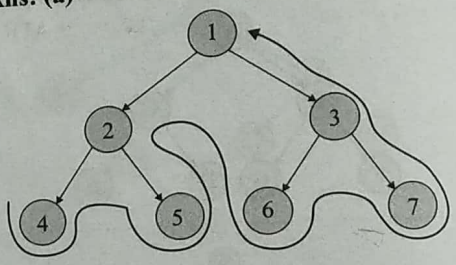
No. of Leaf Nodes = No. of Internal nodes or No. of Internal nodes+1

Internal Nodes = n

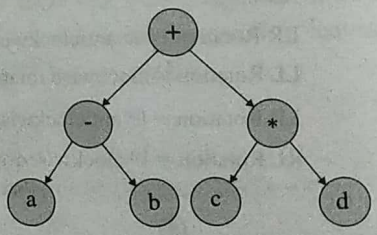
Leaf nodes L = n or n + 1

Total nodes = internal node + leaf node = 2n or 2n+1

18. Ans: (a)



making one-to-one correspondence with we will get the following tree



19. Ans: (a)

Sol: Structurally different trees = Number of Unlabeled binary trees

As per the Catalan Number, the total Number of Unlabeled binary trees

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} = \prod_{k=2}^{n}\frac{n+k}{k} \quad \text{for } n \geq 0$$

For n = 4

the total Number of Unlabeled binary trees

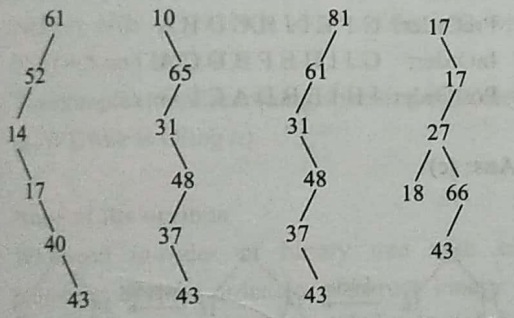
$= 8! /(5! * 4!) = \mathbf{14}$

20. Ans: (d)

Sol: Build Heap or Heap Construction- O(n)
Hash table construction with linear probing - $O(n^2)$ (if all the keys mapped to the same index, each node need to probe over all previously inserted elements)
AVL Tree construction- O(nlogn) (To insert one element in a balanced tree requiered log(n). Therefore construction of AVL tree with n nodes will take O(n* log(n)).)

21. Ans: (d)

Sol: To locate the number 43, we can draw BINARY SEARCH TREE of the given option by keeping in mind there MUST NOT ANY VIOLATION OF Binary Search Tree properties.



Option A Option B Option C Option D

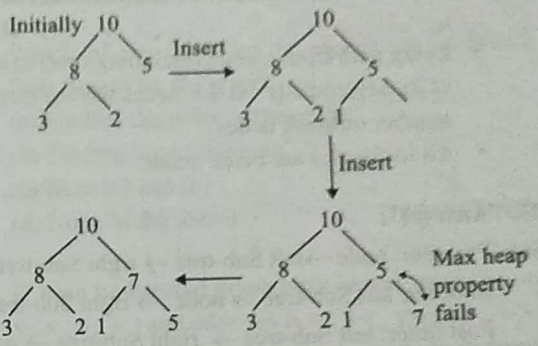
In option d, 18 is to the left of parent node 27. But to locate path we will only traverse to the right hand side of BST.

22. Ans: (d)

Sol: The inorder sequence of a binary search tree always returns the numbers arranged in ascending order. **(Need not to draw BST)**
But as mentioned, The binary search tree uses the reversal ordering on natural numbers i.e. 9 is assumed to be smallest and 0 is assumed to be largest.
So, the inorder sequence will be
9, 8, 7, 6, 5, 4, 3, 2, 1, 0

23. Ans: (d)

Sol:

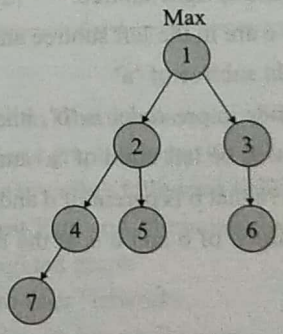


24. Ans: (b)

Sol: In strict binary tree, each node can have either 0 children or 2 children.
L represents Leaf Nodes, I represents Internal Nodes, N represents Total Number Node
L = I + 1(In case of strict binary tree)
N = I + L
N = (L–1) + L = (10–1) + 10 = 19

25. Ans: (b)

Sol:



2nd method
Maximum height of an AVL tree <= 1.44 * $\log_2 n$

26. Ans: (a)

Sol: Every simple path from a node to a descendant leaf contains the same number of black nodes
Red Black Tree Properties:

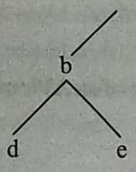
- Every node has a colour either red or black.
- The root of the tree is always black.

- There are no two adjacent red nodes (A red node cannot have a red parent or red child).
- Every path from a node (including root) to any of its descendants' NULL nodes has the same number of black nodes.
- All leaf nodes are black nodes.

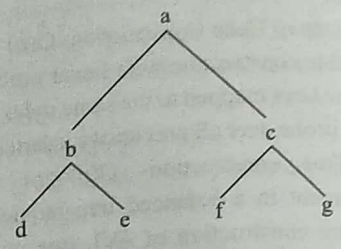
27. Ans: (c)

Sol: Preorder: node →left Sub-tree → right Sub-tree.
In-order: left Sub-tree → node → right Sub-tree.
Post order: left Sub-tree → right Sub-tree → node.
in-order traversal: d b e a f c g
pre-order traversal: a b d e c f g
To construct binary tree from given pre-order and in-order. We traverse pre-order string and find them in in-order string.

1. As per pre-order, a is root of the tree and now we sequentially search 'a' in in-order string. The nodes before 'a' in inorder traversal is in left subtree, nodes after a is right subtree.
   So d b e are in the left subtree and f c g are in the right subtree of 'a'
2. Next node in pre-order is 'b', it is at the left side of a so it will be left child of 'a' and from in-order we can drive that b is parent of d and e where d is at the left subtree of b and e is at the right subtree of 'a'.



3. Next is 'c' in pre-order, according to in-order 'c' will be at the right subtree of a and and c will be parent of f and g where f will be at the left of c and g will be at the right of c.



Post-order of above tree: d e b f g c a

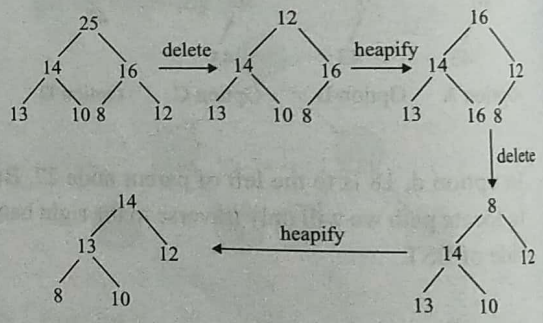
28. None of the options

Tree-1
PreOrder: A B D E G H I J F C
InOrder: B G E H I J D F A C
PostOrder: G J I H E F D B C A

Tree-2
PreOrder: G F E I J H C D B A
InOrder: G J I H E F B D C A
PostOrder: J H I E B D A C F G

29. Ans: (c)

Sol:



Array – 14,13,12,8,10

30. Ans: (c)

Sol: We can find out In-order of given BST: 2,3,4,6,7,9,13,15,17,18,20.
Shortcut-
The inorder sequence of a binary search tree always returns the numbers arranged in ascending order.

17 is just larger than 15 so we can directly right 17

(or)

To find out the in-oder successor of 15, we just search for smallest number of right subtree of 15.

31. **Ans: (c)**

Sol: Important points about AVL Tree

minimum height of AVL tree is floor($\log_2 n$).

If there are n nodes in AVL tree, maximum height can't exceed $1.44*\log_2 n$.

If height of AVL tree is h, maximum number of nodes can be $2^{h+1} - 1$

Minimum number of nodes in a tree with height h can be represented as:

$N(h) = N(h-1) + N(h-2) + 1$ for $n>2$ where $N(0) = 1$ and $N(1) = 2$.

The complexity of searching, inserting and deletion in AVL tree is O(log n).

32. **None of the options**

Sol: We need in-order of binary tree with either pre-order or post-order to construct binary tree uniquely. We have been provided post-order of a binary tree only. So, we cannot create binary tree uniquely with only post-order. So its not possible to determine pre-order.

33. **Ans: (c)**

Sol: Total No. of Nodes =No. of terminal nodes + No. of non-terminal nodes

Max No. of nodes at N^{th} level $= K^N$

Total number of nodes = $(K^{N+1} - 1) / (K-1)$

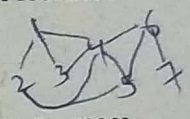
Non Leaf Nodes = $((K^{N+1} - 1) / (K-1)) - K^N$

Non Leaf Nodes / Total Nodes = 1 / K

5. Graphs

01. Which of the following is application of Breath First Search on the graph?
(a) Finding diameter of the graph
(b) Finding bipartite graph
(c) Both (a) and (b)
(d) None of the above **[ISRO - 2018]**

02. G is an undirected graph with vertex set (v_1, v_2, v_3, v_4, v_5, v_6, v_7) and edge set {v_1v_2, v_1v_3, v_1v_4, v_2v_4, v_2v_5, v_3v_4, v_4v_5, v_4v_6, v_5v_6, v_6v_7}. A breadth first search of the graph is performed with v1 as the root node. Which of the following is a tree edge?



(a) v_2v_4 (b) v_1v_4
(c) v_4v_5 (d) v_3v_4

[ISRO - 2020]

KEY & Detailed Solutions	
01. (c)	02. (b) & (c)

01. Ans: (c)

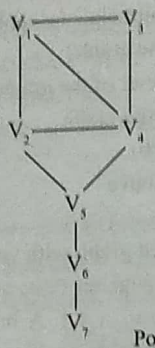
Sol: BFS is used to Find the diameter of the graph and to test whether a graph is bipartite or not.

BFS has many other following applications:

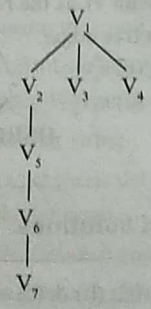
- Shortest Path and Minimum Spanning Tree for unweighted graph
- Peer to Peer Networks:
- Crawlers in Search Engines:
- In social networks, we can find people within a given distance 'k' from a person using Breadth First Search till 'k' levels
- Breadth First Search is used to find all neighboring locations.
  (GPS Navigation systems)
- Broadcasting in Network
- Testing bipartiteness of a graph.
- Implementing parallel algorithms for computing a graph's transitive closure.

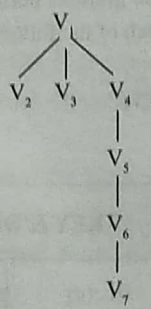
02. Ans: (b) & (c)

Sol: $V_4 V_8$ and $V_1 V_4$



Possible BFS





6. Hashing

01. A symbol table of length 152 is possessing 2 entries at any instant. What is occupation density?
(a) 0.164 (b) 127 (c) 8.06 (d) 6.08
[ISRO - 201

02. Consider a 13 element hash table for which f(key) = key mod 13 is used with integer keys. Assuming linear probing is used for collision resolution, at which location would the key 103 be inserted, if the keys 661, 182, 24 and 103 are inserted in that order?
(a) 0 (b) 1 (c) 11 (d) 12
[ISRO - 2014]

03. A hash table with 10 buckets with one slot per bucket is depicted in fig. The symbols, S1 and S7 are initially entered using a hashing function with linear probing. The maximum number of comparisons needed in searching an item that is not present is

0	S7
1	S1
2	
3	S4
4	S2
5	
6	S5
7	
8	S6
9	S3

(a) 4 (b) 5 (c) 6 (d) 3
[ISRO - 2015]

04. A Hash Function f is defined as f(key) = key mod 7. With linear probing, while inserting the keys 37, 38, 72, 48, 98, 11, 56 into a table indexed from 0, in which location the key 11 will be stored? (count table Index 0 as 0th location)?
(a) 3 (b) 4 (c) 5 (d) 6
[ISRO - 2016]

05. Access time of the symbol table will be logarithmic, if it is implemented by
(a) Linear list (b) Search Tree
(c) Hash Table (d) Self-organization list
[ISRO - 2016]

06. The characters of the string K R P C S N Y T J M are inserted into a hash table of size of size 10 using hash function
h(x) = (ord(x)–ord (A)+1)
If linear probing is used to resolve collisions, then the following insertion causes collision
(a) Y (b) C (c) M (d) P
[ISRO - 2017(Dec)]

07. A hash table with 10 buckets with one slot per bucket is depicted here. The symbols S1 to S7 are initially entered using a hashing function with linear probing. The maximum number of comparisons needed in searching an item that is not present is

0	S7
1	S1
2	
3	S4
4	S2
5	
6	S5
7	
8	S6
9	S3

(a) 4 (b) 5
(c) 6 (d) 3
[ISRO - 2018]

08. In linear hashing, if blocking factor bfr, loading factor i and file buckets N are known, the number of records will be
(a) cr = i + bfr + N (b) r = i – bfr – N
(c) r = i + bfr – N (d) r = i * bfr * N
[ISRO - 2020]

KEY & Detailed Solutions			
01. (a)	02. (b)	03. (b)	04. (c)
05. (b)	06. (c)	07. (b)	

01. Ans: (a)
Sol: Occupation Density = Number of entries / Length of symbol table
= 25 / 152 = 0.164

02. Ans: (b)
Sol:

0	182
1	**103**
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	**661**
12	**24**

03. Ans: (b)
Sol: There are many different cases to find an item that is not present and every may produce different number of comparisons.
Case1: start search from index 0 then total3 comparisons will be made as index 2 found empty.
Case2: start search from index 3 then again total 3 comparisons.
Case3: start search from index 6 then total 2 comparisons.
Case 4: start search from index 8 then comparisons will be made at index 8, 9, 0, 1 and 2 so total 5 comparisons.
So, the correct option is (b).

The maximum number of comparisons needed in searching an item that is not present is size of largest cluster +1.

04. Ans: (c)

Sol: Key → 37, 38, 72, 48, 98, 11, 56

Hash function = f(key) = key mod 7

1st key 37: 37 mod 7 = 2

2nd key 38: 38 mod 7 = 3

3rd key 72: 72 mod 7 = 2, index 2 is occupied by 37 so with linear probing 72 will be inserted at next available space which is 4th index, as 3 is also occupied by 38

4th key 48: 48 mod 7 = 6

5th key 98: 98 mod 7 = 0

6th key 11: 11 mod 7 = 4, but already occupied by 72, so with linear probing it will be inserted at next available space which is 5th index

So, option (c) is correct.

Hash Table

Index	Key
0	98
1	56
2	37
3	38
4	72
5	11
6	48

05. Ans: (b)

Sol: Linear List time complexity to access any element is O(n)

Search Tree time complexity to access any element is O(log n)

Hash Table time complexity to access any element is best case O(1) worst case O(n)

Self organization list time complexity (n+1)/2

06. Ans: (c)

Sol: hash function = h(x) = ((ord(x) – ord(A) + 1)) %
10

that means

if next char of string is A then h(x)
= ((A–A)+1)%10 = 1%10 = 1

if next char of string is B then h(x)
= ((B–A)+1)%10 = 2%10 = 2

if next char of string is C then h(x)
= ((C–A)+1)%10 = 3%10 = 3

…So on

Lets try to insert each char of string in the hash ta of size 10(0-9) with hash function h(x) and lin probing.

Next Char	Hash Value	Collision	Index
K	1	No	1
R	8	No	8
P	6	No	6
C	3	No	3
S	9	No	9
N	4	No	4
Y	5	No	5
T	0	No	0
J	0	Yes	2(next free inde
M	3	Yes	7(next free inde

0	T
1	K
2	J
3	C
4	N
5	Y
6	P
7	M
8	R
9	S

Hash Table

07. **Ans: (b)**

Sol: There are many different cases to find an item that is not present and every may produce different number of comparisons.

Case1: start search from index 0 then total 3 comparisons will be made as index 2 found empty.

Case 2: start search from index 3 then again total 3 comparisons.

Case 3: start search from index 6 then total 2 comparisons.

Case 4: start search from index 8 then comparisons will be made at index 8, 9, 0, 1 and 2 so total 5 comparisons.

So, the correct option is (b).

The maximum number of comparisons needed in searching an item that is not present is size of largest cluster +1.

08. **Ans: (d)**

Sol: Load Factor = Number of keys associated with each bucket

Blocking Factor = Number of records in a block

Total Buckets = (Records/Buckets in Each Block) * (Number of Buckets) = bfr * N

Total Records = Total Number of Buckets * Loading Factor

= i * bfr * N