

## Algorithm and DS MCQ

**1. Which of this best describes an array?**

- a) A data structure that shows a hierarchical behaviour
- b) Container of objects of similar types**
- c) Arrays are immutable once initialised
- d) Array is not a data structure

**2. How do you initialize an array in C?**

- a) `int arr[3] = (1,2,3);`
- b) `int arr(3) = {1,2,3};`
- c) `int arr[3] = {1,2,3};`**
- d) `int arr(3) = (1,2,3);`

**3. When does the `ArrayIndexOutOfBoundsException` occur?**

- a) Compile-time
- b) Run-time**
- c) Not an error
- d) Not an exception at all

**4. Which of the following concepts make extensive use of arrays?**

- a) Binary trees
- b) Scheduling of processes
- c) Caching
- d) Spatial locality**

**5. What are the advantages of arrays?**

- a) Objects of mixed data types can be stored
- b) Elements in an array cannot be sorted
- c) Index of first element of an array is 1
- d) Easier to store elements of same data type**

**6. What are the disadvantages of arrays?**

- a) Data structure like queue or stack cannot be implemented
- b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size**
- c) Index value of an array can be negative
- d) Elements are sequentially accessed

**7. Assuming int is of 4bytes, what is the size of `int arr[15];`?**

- a) 15
- b) 19
- c) 11
- d) 60**

8. In general, the index of the first element in an array is \_\_\_\_\_

- a) 0
- b) -1
- c) 2
- d) 1

9. Elements in an array are accessed \_\_\_\_\_

- a) randomly
- b) sequentially
- c) exponentially
- d) logarithmically

10. Process of inserting an element in stack is called \_\_\_\_\_

- a) Create
- b) Push
- c) Evaluation
- d) Pop

11. Process of removing an element from stack is called \_\_\_\_\_

- a) Create
- b) Push
- c) Evaluation
- d) Pop

12. In a stack, if a user tries to remove an element from an empty stack it is called \_\_\_\_\_

- a) Underflow
- b) Empty collection
- c) Overflow
- d) Garbage Collection

13. Pushing an element into stack already having five elements and stack size of 5, then stack becomes \_\_\_\_\_

- a) Overflow
- b) Crash
- c) Underflow
- d) User flow

14. Entries in a stack are "ordered". What is the meaning of this statement?

- a) A collection of stacks is sortable
- b) Stack entries may be compared with the '<' operation
- c) The entries are stored in a linked list
- d) There is a Sequential entry that is one by one

**15. Which of the following is not the application of stack?**

- a) A parentheses balancing program
- b) Tracking of local variables at run time
- c) Compiler Syntax Analyzer
- d) Data Transfer between two asynchronous process**

**16. Consider the usual algorithm for determining whether a sequence of parentheses is balanced. The maximum number of parentheses that appear on the stack AT ANY ONE TIME when the algorithm analyzes:  $((()())())$ ?**

- a) 1
- b) 2
- c) 3**
- d) 4 or more

**17. Consider the usual algorithm for determining whether a sequence of parentheses is balanced. Suppose that you run the algorithm on a sequence that contains 2 left parentheses and 3 right parentheses (in some order). The maximum number of parentheses that appear on the stack AT ANY ONE TIME during the computation?**

- a) 1
- b) 2**
- c) 3
- d) 4 or more

**18. What is the value of the postfix expression  $6\ 3\ 2\ 4\ +\ -\ *$ ?**

- a) 1
- b) 40
- c) 74
- d) -18**

**19. Here is an infix expression:  $4 + 3*(6*3-12)$ . Suppose that we are using the usual stack algorithm to convert the expression from infix to postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression?**

- a) 1
- b) 2
- c) 3
- d) 4**

**20. The postfix form of the expression  $(A + B)*(C*D - E)*F / G$  is?**

- a)  $AB + CD * E - FG / **$
- b)  $AB + CD * E - F ** G /$
- c)  $AB + CD * E - * F * G /$**
- d)  $AB + CDE * - * F * G /$

- 21. The data structure required to check whether an expression contains a balanced parenthesis is?**
- a) **Stack**
  - b) Queue
  - c) Array
  - d) Tree
- 22. What data structure would you mostly likely see in non recursive implementation of a recursive algorithm?**
- a) Linked List
  - b) **Stack**
  - c) Queue
  - d) Tree
- 23. The process of accessing data stored in a serial access memory is similar to manipulating data on a \_\_\_\_\_**
- a) Heap
  - b) Binary Tree
  - c) Array
  - d) **Stack**
- 24. The postfix form of  $A*B+C/D$  is?**
- a)  $*AB/CD+$
  - b)  **$AB*CD/+$**
  - c)  $A*BC+/D$
  - d)  $ABCD+/*$
- 25. Which data structure is needed to convert infix notation to postfix notation?**
- a) Branch
  - b) Tree
  - c) Queue
  - d) **Stack**
- 26. The prefix form of  $A-B/(C * D ^ E)$  is?**
- a)  $-/*^ACBDE$
  - b)  $-ABCD*^DE$
  - c)  **$-A/B*C^DE$**
  - d)  $-A/BC*^DE$

**27. What is the result of the following operation?**

**Top (Push (S, X))**

- a) X**
- b) X+S
- c) S
- d) XS

**28. The prefix form of an infix expression  $(p + q) - (r * t)$  is?**

- a)  $+ pq - *rt$
- b)  $- +pqr * t$
- c)  $- +pq * rt$**
- d)  $- + * pqrt$

**29. Which data structure is used for implementing recursion?**

- a) Queue
- b) Stack**
- c) Array
- d) List

**30. The result of evaluating the postfix expression 5, 4, 6, +, \*, 4, 9, 3, /, +, \* is?**

- a) 600
- b) 350**
- c) 650
- d) 588

**31. Convert the following infix expressions into its equivalent postfix expressions.**

**$(A + B \wedge D)/(E - F) + G$**

- a)  $(A B D \wedge + E F - / G +)$**
- b)  $(A B D + \wedge E F - / G +)$
- c)  $(A B D \wedge + E F / - G +)$
- d)  $(A B D E F + \wedge / - G +)$

**32. Convert the following Infix expression to Postfix form using a stack.**

**$x + y * z + (p * q + r) * s$ , Follow usual precedence rule and assume that the expression is legal.**

- a)  $xyz*+pq*r+s*+$**
- b)  $xyz*+pq*r+s+*$
- c)  $xyz+*pq*r+s*+$
- d)  $xyzp+**qr+s*+$

**33. Which of the following statement(s) about stack data structure is/are NOT correct?**

- a) Linked List are used for implementing Stacks
- b) Top of the Stack always contain the new node
- c) Stack is the FIFO data structure**
- d) Null link is present in the last node at the bottom of the stack

**34. Consider the following operation performed on a stack of size 5.**

```
Push(1);  
Pop();  
Push(2);  
Push(3);  
Pop();  
Push(4);  
Pop();  
Pop();  
Push(5);
```

**After the completion of all operation, the number of elements present in stack is?**

- a) 1**
- b) 2
- c) 3
- d) 4

**35. Which of the following is not an inherent application of stack?**

- a) Reversing a string
- b) Evaluation of postfix expression
- c) Implementation of recursion
- d) Job scheduling**

**36. The type of expression in which operator succeeds its operands is?**

- a) Infix Expression
- b) Prefix Expression
- c) Postfix Expression**
- d) Both Prefix and Postfix Expressions

**37. 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 for the infix expression  $a + b \times c - d^e^f$  is?**

- a)  $abc \times + def^{^^} -$
- b)  $abc \times + de^{^f^} -$**
- c)  $ab+c \times d - e^{^f^}$
- d)  $-+a \times bc^{^}^{^} def$

- 38. If the elements "A", "B", "C" and "D" are placed in a stack and are deleted one at a time, what is the order of removal?**
- a) ABCD
  - b) DCBA**
  - c) DCAB
  - d) ABDC
- 39. A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as \_\_\_\_\_**
- a) Queue**
  - b) Stack
  - c) Tree
  - d) Linked list
- 40. The data structure required for Breadth First Traversal on a graph is?**
- a) Stack
  - b) Array
  - c) Queue**
  - d) Tree
- 41. A queue follows \_\_\_\_\_**
- a) FIFO (First In First Out) principle**
  - b) LIFO (Last In First Out) principle
  - c) Ordered array
  - d) Linear tree
- 42. Circular Queue is also known as \_\_\_\_\_**
- a) Ring Buffer**
  - b) Square Buffer
  - c) Rectangle Buffer
  - d) Curve Buffer
- 43. If the elements "A", "B", "C" and "D" are placed in a queue and are deleted one at a time, in what order will they be removed?**
- a) ABCD**
  - b) DCBA
  - c) DCAB
  - d) ABDC

- 44. A data structure in which elements can be inserted or deleted at/from both ends but not in the middle is?**
- a) Queue
  - b) Circular queue
  - c) Dequeue**
  - d) Priority queue
- 45. A normal queue, if implemented using an array of size MAX\_SIZE, gets full when?**
- a) Rear = MAX\_SIZE - 1**
  - b) Front = (rear + 1) mod MAX\_SIZE
  - c) Front = rear + 1
  - d) Rear = front
- 46. Queues serve major role in \_\_\_\_\_**
- a) Simulation of recursion
  - b) Simulation of arbitrary linked list
  - c) Simulation of limited resource allocation**
  - d) Simulation of heap sort
- 47. Which of the following is not the type of queue?**
- a) Ordinary queue
  - b) Single ended queue**
  - c) Circular queue
  - d) Priority queue
- 48. A linear collection of data elements where the linear node is given by means of pointer is called?**
- a) Linked list**
  - b) Node list
  - c) Primitive list
  - d) Unordered list
- 49. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?**
- i) Insertion at the front of the linked list
  - ii) Insertion at the end of the linked list
  - iii) Deletion of the front node of the linked list
  - iv) Deletion of the last node of the linked list
- a) I and II
  - b) I and III**
  - c) I, II and III
  - d) I, II and IV



- 50. In linked list each node contains a minimum of two fields. One field is data field to store the data second field is?**
- a) Pointer to character
  - b) Pointer to integer
  - c) Pointer to node**
  - d) Node
- 51. What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?**
- a)  $O(1)$
  - b)  $O(n)$
  - c)  $\theta(n)$**
  - d)  $\theta(1)$
- 52. What would be the asymptotic time complexity to insert an element at the front of the linked list (head is known)?**
- a)  $O(1)$**
  - b)  $O(n)$
  - c)  $O(n^2)$
  - d)  $O(n^3)$
- 53. What would be the asymptotic time complexity to find an element in the linked list?**
- a)  $O(1)$
  - b)  $O(n)$**
  - c)  $O(n^2)$
  - d)  $O(n^4)$
- 54. What would be the asymptotic time complexity to insert an element at the second position in the linked list?**
- a)  $O(1)$**
  - b)  $O(n)$**
  - c)  $O(n^2)$**
  - d)  $O(n^3)$**
- 55. The concatenation of two lists can be performed in  $O(1)$  time. Which of the following variation of the linked list can be used?**
- a) Singly linked list
  - b) Doubly linked list
  - c) Circular doubly linked list**
  - d) Array implementation of list

**56. Consider the following definition in c programming language.**

```
struct node
{
    int data;
    struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

**Which of the following c code is used to create new node?**

- a) ptr = (NODE\*)malloc(sizeof(NODE));**
- b) ptr = (NODE\*)malloc(NODE);
- c) ptr = (NODE\*)malloc(sizeof(NODE\*));
- d) ptr = (NODE)malloc(sizeof(NODE));

**57. What kind of linked list is best to answer questions like "What is the item at position n?"**

- a) Singly linked list
- b) Doubly linked list
- c) Circular linked list
- d) Array implementation of linked list**

**58. Linked lists are not suitable for the implementation of \_\_\_\_\_**

- a) Insertion sort
- b) Radix sort
- c) Polynomial manipulation
- d) Binary search**

**59. Linked list is considered as an example of \_\_\_\_\_ type of memory allocation.**

- a) Dynamic**
- b) Static
- c) Compile time
- d) Heap

**60. In Linked List implementation, a node carries information regarding \_\_\_\_\_**

- a) Data
- b) Link
- c) Data and Link**
- d) Node

**61. Linked list data structure offers considerable saving in \_\_\_\_\_**

- a) Computational Time
- b) Space Utilization
- c) Space Utilization and Computational Time**
- d) Speed Utilization

**62. Which of the following points is/are not true about Linked List data structure when it is compared with an array?**

- a) Arrays have better cache locality that can make them better in terms of performance
- b) It is easy to insert and delete elements in Linked List
- c) Random access is not allowed in a typical implementation of Linked Lists
- d) Access of elements in linked list takes less time than compared to arrays**

**63. What does the following function do for a given Linked List with first node as head?**

```
void fun1(struct node* head)
{
    if(head == NULL)
        return;
    fun1(head->next);
    printf("%d ", head->data);
}
```

- a) Prints all nodes of linked lists
- b) Prints all nodes of linked list in reverse order**
- c) Prints alternate nodes of Linked List
- d) Prints alternate nodes in reverse order

**64. Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?**

- a) Insertion Sort
- b) Quick Sort
- c) Heap Sort
- d) Merge Sort**

**65. Which of the following is not a linear data structure**

- a) Array
- b) Linked List
- c) Stack
- d) Tree**

**66. The complexity of linear search algorithm is**

- a)  $O(\log n)$
- b)  $O(n)$**
- c)  $O(n^2)$
- d)  $O(n \log n)$

**67. A Complete Binary tree is defined as binary tree where?**

- a) All leaf. nodes are on level  $n+1$  and  $n$
- b) All leaf nodes are on level  $n$
- c) All leaf nodes are on level  $n-1$
- d) All leaf nodes are on level  $n$  and  $n-1$**

68. The infix expression  $5 + 3 * 9 / (7 - 4) - 6 * 2$  when converted to postfix the final result would evaluate to?
- a) 2
  - b) 1
  - c) 3
  - d) 4
69. What will be the output of following expression when we convert infix to postfix?  
 $((A+B) * (C-D)) / E$
- a)  $AB + CD - E * /$
  - b)  $AB + CD * - E /$
  - c)  $AB + CD * E - /$
  - d)  **$AB + CD - * E /$**
70. Worst Case Complexity of Insertion sort?
- a)  **$O(n^2)$**
  - b)  $O(n)$
  - c)  $O(1)$
  - d)  $O(n \log n)$
71. Construct a Binary Search Tree and give the preorder Traversal  
20,30,10,5,16,21,29,45,0,15,6 ?
- a) 20 10 5 0 6 16 15 30 21 45 29
  - b) 20 10 5 6 0 30 21 29 45 29 45
  - c) 20 10 5 0 6 16 15 29 21 45 30
  - d) **20 10 5 0 6 16 15 30 21 29 45**
72. Which of the following algorithm does not uses divide and conquer strategy?
- a) Quick Sort
  - b) Merge Sort
  - c) **Binary search**
  - d) All of the above
73. The operation of visiting each element exactly once in the list is known as?
- a) Sorting
  - b) Merging
  - c) Inserting
  - d) **Traversal**
74. Which of the following is not a collision resolution technique in hashing?
- a) Open addressing
  - b) Separate chaining
  - c) Probing
  - d) **Polling**

**75. A Complete Graph with N vertices has?**

- a) n edges
- b) n-1 edges
- c)  $n(n-1)/2$  edges**
- d) n(n-1) edges

**76. Which of the following condition checks the overflow condition for stack?**

- a)  $\text{Top} = \text{MAX} + 1$
- b)  $\text{Top} = \text{MAX} - 1$**
- c)  $\text{Top} = -1$
- d) All of the above

**77. Depth first search can be implemented using**

- a) Queue
- b) Stack**
- c) Linked list
- d) Array

**78. Which of the following sorting algorithm is also called as partition exchange sort?**

- a) Insertion Sort
- b) Selection Sort
- c) Radix Sort
- d) Quick Sort**

**79. How many cycles does spanning tree have?**

- a) Minimum 1
- b) Greater than 1
- c) 0**
- d) 1

**80. Which of the following structure is used to implement linked lists in C?**

- a) `typedef struct node { int data; struct node *next; }NODE;`**
- b) `typedef struct node { int data; node *next; }NODE;`
- c) `typedef struct node { int data; struct node next; }*NODE;`
- d) Both 1 and 2

**81. Which of the following Data Structures are implemented in cut,copy,paste operations?**

- a) Queue
- b) Stack
- c) Both of Above**
- d) Graph

82. A graph is a collection of nodes, called \_\_\_\_\_ And line segments called arcs or \_\_\_\_\_ that connect pair of nodes.

- a) vertices, paths
- b) vertices, edges**
- c) graph node, edges
- d) edges, vertices

83. A hash function  $h$  defined  $h(\text{key}) = \text{key} \bmod 7$ , with linear probing, is used to insert the keys 44,45,79,55,91,18,63 into a table indexed from 0 to 6.

What will be the location of key 18?

- a) 3
- b) 4
- c) 5**
- d) 6

84. Dijkstra's Algorithm is used to solve \_\_\_\_\_ problems.

- a) All pair shortest path
- b) Network flow
- c) Single source shortest path**
- d) Sorting

85. The concatenation of two list can performed in  $O(1)$  time. Which of the following variation of linked list can be used?

- a) Singly linked list
- b) Doubly linked list
- c) Circular doubly linked list**
- d) Array implementation of list

86. What is the speciality about the inorder traversal of a binary search tree?

- a) It traverses in a non increasing order
- b) It traverses in an increasing order**
- c) It traverses in a random fashion
- d) It traverses based on priority of the node

87. What will be the height of a balanced full binary tree with 8 leaves?

- a) 8
- b) 5
- c) 6
- d) 4**

88. Running merge sort on an array of size  $n$  which is already sorted is

- a)  $O(n)$
- b)  $O(n \log n)$**
- c)  $O(n^2)$
- d) None

**89. Quadratic probing overcomes primary collision.**

- a) **True**
- b) False

**90. The post order traversal of binary tree is DEBFCA. Find out the pre order traversal.**

- a) ABFCDE
- b) ADBFEC
- c) **ABDECF**
- d) ABDCEF

**91. Breadth First search**

- a) **scans all incident edges before moving to other vertex**
- b) Scans adjacent unvisited vertex as soon as possible
- c) Is same as backtracking
- d) Computer a path between two vertices of graph equivalently

**92. The result of evaluating prefix expression  $*/b+-dacd$ , where  $a=3$ ,  $b=6$ ,  $c=1$ ,  $d=5$  is:**

- a) 0
- b) 5
- c) **10**
- d) 15

**93.  $O(n^2)$  means computing time is \_\_\_\_\_**

- a) Constant
- b) **Quadratic**
- c) Linear
- d) Cubic

**94. The number of edges from the node to the deepest leaf is called \_\_\_\_\_ of the tree.**

- a) **Height**
- b) Depth
- c) Length
- d) None of the mentioned

**95. A circular queue of size N will sign queue full when the number of elements in the queue is**

- a) **N-1**
- b) N
- c) N+1
- d) N-2

**96. Which of the following operations is performed more efficiently by doubly linked list than by singly linked list?**

- a) **Deleting a node whose location is given**
- b) Searching of an unsorted list for a given item
- c) Inverting a node after the node with given location
- d) Traversing a list to process each node

**97. Consider the following doubly linear linked list and find the output of given code:**

```
head                                tail
1 <--> 2 <--> 3 <--> 4 <--> 5
4000 2000 2800 4800 3000
trav= tail;
while(trav!=NULL && trav->prev!=NULL)
{
    print("%d-->",trav->data);
    trav = trav->prev->prev;
}
```

- a) 5-->3-->1
- b) 5-->4-->3-->2-->1
- c) 5-->3-->--A
- d) 1-->3-->5

**98. The following postfix expression with single digit operands is evaluated using a stack:**

**8 2 3 ^ / 2 3 \* + 5 1 \* -**

**Note that ^ is the exponentiation operator. The top two elements of the stack after the first \* is \_\_\_\_\_**

- a) **6,1**
- b) 5,7
- c) 3,2
- d) 1,5



99. The following function reverse () is supposed to reverse a singly linked list. There is one line missing at the end of the function.

```
/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* head_ref is a double pointer which points to head (or start) pointer
of linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}
```

What should be added in place of "/\*ADD A STATEMENT HERE\*/", so that the function correctly reverses a linked list.

- a) \*head\_ref = prev;
- b) \*head\_ref = current;
- c) \*head\_ref = next;
- d) \*head\_ref = NULL;

100. What is the output of following function for start pointing to first node of following linked list?

```
1->2->3->4->5->6
void fun(struct node* start)
{
    if(start == NULL)
        return;
    printf("%d ", start->data);
    if(start->next != NULL )
        fun(start->next->next);
    printf("%d ", start->data);
}
```

- a) 1 4 6 6 4 1
- b) 1 3 5 1 3 5
- c) 1 2 3 5
- d) 1 3 5 5 3 1

**101. The following C function takes a simply-linked list as an input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.**

```
typedef struct node
{
    int value;
    struct node *next;
}Node;

Node *move_to_front(Node *head)
{
    Node *p, *q;
    if ((head == NULL) || (head->next == NULL))
        return head;
    q = NULL; p = head;
    while (p->next !=NULL)
    {
        q = p;
        p = p->next;
    }
    _____
    return head;
}
```

- a) q = NULL; p->next = head; head = p;
- b) q->next = NULL; head = p; p->next = head;
- c) head = p; p->next = q; q->next = NULL;
- d) q->next = NULL; p->next = head; head = p;**

**102. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?**

```
struct node {
    int value;
    struct node *next;
};

void rearrange(struct node *list){
    struct node *p, * q;
    int temp;
    if ((!list) || !list->next)
        return;
    p = list;
    q = list->next;
    while(q) {
        temp = p->value;
        p->value = q->value;
        q->value = temp;
        p = q->next;
        q = p?p->next:0;
    }
}
```

- a) 1, 2, 3, 4, 5, 6, 7
- b) 2, 1, 4, 3, 6, 5, 7**
- c) 1, 3, 2, 5, 4, 7, 6
- d) 2, 3, 4, 5, 6, 7, 1

- 103. In the worst case, the number of comparisons needed to search a singly linked list of length  $n$  for a given element is?**
- a)  $\log_2 n$
  - b)  $n/2$
  - c)  $\log_2 n - 1$
  - d)  $n$**
- 104. Given pointer to a node  $X$  in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node  $X$  from given linked list?**
- a) Possible if  $X$  is not last node**
  - b) Possible if size of linked list is even
  - c) Possible if size of linked list is odd
  - d) Possible if  $X$  is not first node
- 105. You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?**
- a) Delete the first element
  - b) Insert a new element as a first element
  - c) Delete the last element of the list**
  - d) Add a new element at the end of the list
- 106. Which of the following is not a disadvantage to the usage of array?**
- a) Fixed size
  - b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
  - c) Insertion based on position
  - d) Accessing elements at specified positions**
- 107. What is the time complexity of inserting at the end in dynamic arrays?**
- a)  $O(1)$
  - b)  $O(n)$
  - c)  $O(\log n)$
  - d) Either  $O(1)$  or  $O(n)$**
- 108. What is the time complexity to count the number of elements in the linked list?**
- a)  $O(1)$
  - b)  $O(n)$**
  - c)  $O(\log n)$
  - d)  $O(n^2)$

**109. What is the space complexity for deleting a linked list?**

- a)  $O(1)$
- b)  $O(n)$
- c) Either  $O(1)$  or  $O(n)$
- d)  $O(\log n)$

**110. Which of these is not an application of a linked list?**

- a) To implement file systems
- b) For separate chaining in hash-tables
- c) To implement non-binary trees
- d) **Random Access of elements**

**111. Which of the following is false about a doubly linked list?**

- a) We can navigate in both the directions
- b) It requires more space than a singly linked list
- c) The insertion and deletion of a node take a bit longer
- d) **Implementing a doubly linked list is easier than singly linked list**

**112. What is a memory efficient double linked list?**

- a) **Each node has only one pointer to traverse the list back and forth**
- b) The list has breakpoints for faster traversal
- c) An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list
- d) A doubly linked list that uses bitwise AND operator for storing addresses

**113. How do you calculate the pointer difference in a memory efficient double linked list?**

- a) head xor tail
- b) **pointer to previous node xor pointer to next node**
- c) pointer to previous node – pointer to next node
- d) pointer to next node – pointer to previous node

**114. What is the worst case time complexity of inserting a node in a doubly linked list?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  **$O(n)$**
- d)  $O(1)$

**115. What differentiates a circular linked list from a normal linked list?**

- a) You cannot have the 'next' pointer point to null in a circular linked list
- b) It is faster to traverse the circular linked list
- c) **You may or may not have the 'next' pointer point to null in a circular linked list**
- d) Head node is known in circular linked list

**116. What is the time complexity of searching for an element in a circular linked list?**

- a)  **$O(n)$**
- b)  $O(n \log n)$
- c)  $O(1)$
- d)  $O(n^2)$

**117. Which of the following application makes use of a circular linked list?**

- a) Undo operation in a text editor
- b) Recursive function calls
- c) **Allocating CPU to resources**
- d) Implement Hash Tables

**118. Which of the following is false about a circular linked list?**

- a) Every node has a successor
- b) **Time complexity of inserting a new node at the head of the list is  $O(1)$**
- c) Time complexity for deleting the last node is  $O(n)$
- d) We can traverse the whole circular linked list by starting from any point

**119. Consider a small circular linked list. How to detect the presence of cycles in this list effectively?**

- a) Keep one node as head and traverse another temp node till the end to check if its 'next' points to head
- b) **Have fast and slow pointers with the fast pointer advancing two nodes at a time and slow pointer advancing by one node at a time**
- c) Cannot determine, you have to pre-define if the list contains cycles
- d) Circular linked list itself represents a cycle. So no new cycles cannot be generated

**120. Which of the following real world scenarios would you associate with a stack data structure?**

- a) **piling up of chairs one above the other**
- b) people standing in a line to be serviced at a counter
- c) offer services based on the priority of the customer
- d) tatkal Ticket Booking in IRCTC

**121. What does the following function check for? (all necessary headers to be included and function is called from main)**

```
#define MAX 10

typedef struct stack
{
    int top;
    int item[MAX];
}stack;

int function(stack *s)
{
    if(s->top == -1)
        return 1;
    else return 0;
}
```

- a) full stack
- b) invalid index
- c) empty stack**
- d) infinite stack

**122. What does 'stack underflow' refer to?**

- a) accessing item from an undefined stack
- b) adding items to a full stack
- c) removing items from an empty stack**
- d) index out of bounds exception

**123. What is the time complexity of pop() operation when the stack is implemented using an array?**

- a) O(1)**
- b) O(n)
- c) O(logn)
- d) O(nlogn)

**124. Which of the following array position will be occupied by a new element being pushed for a stack of size N elements(capacity of stack > N)?**

- a) S[N-1]
- b) S[N]**
- c) S[1]
- d) S[0]

**125. Array implementation of Stack is not dynamic, which of the following statements supports this argument?**

- a) space allocation for array is fixed and cannot be changed during run-time**
- b) user unable to give the input for stack operations
- c) a runtime exception halts execution
- d) improper program compilation

- 126. Which of the following array element will return the top-of-the-stack-element for a stack of size N elements(capacity of stack > N)?**
- a) **S[N-1]**
  - b) S[N]
  - c) S[N-2]
  - d) S[N+1]
- 127. What is the best case time complexity of deleting a node in a Singly Linked list?**
- a) O (n)
  - b) O (n<sup>2</sup>)
  - c) O (nlogn)
  - d) **O (1)**
- 128. Which of the following statements are not correct with respect to Singly Linked List(SLL) and Doubly Linked List(DLL)?**
- a) Complexity of Insertion and Deletion at known position is O(n) in SLL and O(1) in DLL
  - b) SLL uses lesser memory per node than DLL
  - c) DLL has more searching power than SLL
  - d) **Number of node fields in SLL is more than DLL**
- 129. What does 'stack overflow' refer to?**
- a) accessing item from an undefined stack
  - b) **adding items to a full stack**
  - c) removing items from an empty stack
  - d) index out of bounds exception
- 130. Which of the following data structures can be used for parentheses matching?**
- a) n-ary tree
  - b) queue
  - c) priority queue
  - d) **stack**
- 131. Minimum number of queues to implement stack is \_\_\_\_\_**
- a) 3
  - b) 4
  - c) **1**
  - d) 2
- 132. Which of the following properties is associated with a queue?**
- a) First In Last Out
  - b) **First In First Out**
  - c) Last In First Out
  - d) Last In Last Out

**133. In a circular queue, how do you increment the rear end of the queue?**

- a) rear++
- b) (rear+1) % CAPACITY**
- c) (rear % CAPACITY)+1
- d) rear-

**134. What is the term for inserting into a full queue known as?**

- a) overflow**
- b) underflow
- c) null pointer exception
- d) program won't be compiled

**135. What is the time complexity of enqueue operation?**

- a)  $O(\log n)$
- b)  $O(n \log n)$
- c)  $O(n)$
- d)  $O(1)$**

**136. What is the need for a circular queue?**

- a) effective usage of memory**
- b) easier computations
- c) to delete elements based on priority
- d) implement LIFO principle in queues

**137. What is the space complexity of a linear queue having n elements?**

- a)  $O(n)$**
- b)  $O(n \log n)$
- c)  $O(\log n)$
- d)  $O(1)$

**138. In linked list implementation of queue, if only front pointer is maintained, which of the following operation take worst case linear time?**

- a) Insertion
- b) Deletion
- c) To empty a queue
- d) Both Insertion and To empty a queue**

**139. In linked list implementation of a queue, where does a new element be inserted?**

- a) At the head of link list
- b) At the centre position in the link list
- c) At the tail of the link list**
- d) At any position in the linked list



- 140. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a NONEMPTY queue?**
- a) Only front pointer
  - b) Only rear pointer**
  - c) Both front and rear pointer
  - d) No pointer will be changed
- 141. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into EMPTY queue?**
- a) Only front pointer
  - b) Only rear pointer
  - c) Both front and rear pointer**
  - d) No pointer will be changed
- 142. In case of insertion into a linked queue, a node borrowed from the \_\_\_\_\_ list is inserted in the queue.**
- a) AVAIL**
  - b) FRONT**
  - c) REAR**
  - d) NULL**
- 143. In linked list implementation of a queue, from where is the item deleted?**
- a) At the head of link list**
  - b) At the centre position in the link list
  - c) At the tail of the link list
  - d) Node before the tail
- 144. In linked list implementation of a queue, the important condition for a queue to be empty is?**
- a) FRONT is null**
  - b) REAR is null
  - c) LINK is empty
  - d)  $FRONT == REAR - 1$
- 145. The essential condition which is checked before insertion in a linked queue is?**
- a) Underflow
  - b) Overflow**
  - c) Front value
  - d) Rear value

**146. The essential condition which is checked before deletion in a linked queue is?**

- a) Underflow
- b) Overflow
- c) Front value
- d) Rear value

**147. Which of the following is true about linked list implementation of queue?**

- a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end
- b) In push operation, if new nodes are inserted at the beginning, then in pop operation, nodes must be removed from the beginning
- c) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from end
- d) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from beginning

**148. With what data structure can a priority queue be implemented?**

- a) Array
- b) List
- c) Heap
- d) Tree

**149. Which of the following is not an application of priority queue?**

- a) Huffman codes
- b) Interrupt handling in operating system
- c) Undo operation in text editors
- d) Bayesian spam filter

**150. What is the time complexity to insert a node based on key in a priority queue?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$

**151. What is not a disadvantage of priority scheduling in operating systems?**

- a) A low priority process might have to wait indefinitely for the CPU
- b) If the system crashes, the low priority systems may be lost permanently
- c) Interrupt handling
- d) Indefinite blocking

**152. Which of the following is not an advantage of a priority queue?**

- a) Easy to implement
- b) Processes with different priority can be efficiently handled
- c) Applications with differing requirements
- d) Easy to delete elements in any case**

**153. What is the time complexity to insert a node based on position in a priority queue?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$**
- d)  $O(n^2)$

**154. What is a dequeue?**

- a) A queue with insert/delete defined for both front and rear ends of the queue**
- b) A queue implemented with a doubly linked list
- c) A queue implemented with both singly and doubly linked lists
- d) A queue with insert/delete defined for front side of the queue

**155. What are the applications of dequeue?**

- a) A-Steal job scheduling algorithm
- b) Can be used as both stack and queue
- c) To find the maximum of all sub arrays of size k
- d) To avoid collision in hash tables**

**156. What is the time complexity of deleting from the rear end of the dequeue implemented with a singly linked list?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$**
- d)  $O(n^2)$

**157. After performing these set of operations, what does the final list look contain?**

```
InsertFront (10);  
InsertFront (20);  
InsertRear (30);  
DeleteFront ();  
InsertRear (40);  
InsertRear (10);  
DeleteRear ();  
InsertRear (15);  
display ();
```

- a) 10 30 10 15
- b) 20 30 40 15
- c) 20 30 40 10
- d) 10 30 40 15**

**158. How many stacks are required for applying evaluation of infix expression algorithm?**

- a) one
- b) two**
- c) three
- d) four

**159. How many passes does the evaluation of infix expression algorithm makes through the input?**

- a) One**
- b) Two
- c) Three
- d) Four

**160. Identify the infix expression from the list of options given below.**

- a)  $a/b+(c-d)$**
- b)  $abc^*+d+ab+cd+^*ce-f-$
- c)  $ab-c-$
- d)  $+ab$

**161. Which of the following statement is incorrect with respect to evaluation of infix expression algorithm?**

- a) Operand is pushed on to the stack
- b) If the precedence of operator is higher, pop two operands and evaluate**
- c) If the precedence of operator is lower, pop two operands and evaluate
- d) The result is pushed on to the operand stack

**162. Evaluate the following statement using infix evaluation algorithm and choose the correct answer.  $1+2*3-2$**

- a) 3
- b) 6
- c) 5**
- d) 4

**163. Evaluation of infix expression is done based on precedence of operators.**

- a) True**
- b) False

**164. Of the following choices, which operator has the lowest precedence?**

- a)  $\wedge$
- b)  $+$
- c)  $/$
- d)  $\#$**

**165. The system throws an error if parentheses are encountered in an infix expression evaluation algorithm.**

- a) True
- b) False**

**166. Evaluate the following and choose the correct answer.**

**$a/b+c*d$  where  $a=4, b=2, c=2, d=1$ .**

- a) 1
- b) 4**
- c) 5
- d) 2

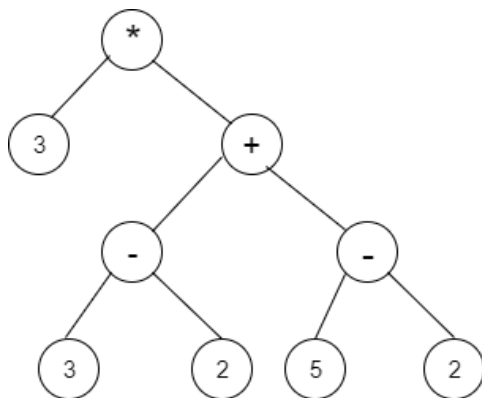
**167. Evaluate the following statement using infix evaluation algorithm and choose the correct answer.  $4*2+3-5/5$**

- a) 10
- b) 11
- c) 16
- d) 12**

**168. Evaluate the following infix expression using algorithm and choose the correct answer.  $a+b*c-d/e^f$  where  $a=1, b=2, c=3, d=4, e=2, f=2$ .**

- a) 6**
- b) 8
- c) 9
- d) 7

**169. From the given expression tree, identify the infix expression, evaluate it and choose the correct result.**



- a) 5
- b) 10
- c) 12**
- d) 16

**170. How many stacks are required for evaluation of prefix expression?**

- a) one
- b) two**
- c) three
- d) four

**171. While evaluating a prefix expression, the string is read from?**

- a) left to right
- b) right to left**
- c) center to right
- d) center to left to right

**172. The associativity of an exponentiation operator  $^$  is right side.**

- a) True**
- b) False

**173. How many types of input characters are accepted by this algorithm?**

- a) one
- b) two
- c) three**
- d) four

**174. What determines the order of evaluation of a prefix expression?**

- a) precedence and associativity**
- b) precedence only
- c) associativity only
- d) depends on the parser

**175. Find the output of the following prefix expression.**

$*+2-2\ 1/-4\ 2+-5\ 3\ 1$

- a) 2**
- b) 12
- c) 10
- d) 4

**176. An error is thrown if the character ' $\backslash n$ ' is pushed in to the character stack.**

- a) true
- b) false**

**177. Using the evaluation of prefix algorithm, evaluate + - 9 2 7.**

- a) 10
- b) 4
- c) 17
- d) 14**

**178. If  $- * + a b c d = 11$ , find a, b, c, d using evaluation of prefix algorithm.**

- a) a=2, b=3, c=5, d=4
- b) a=1, b=2, c=5, d=4**
- c) a=5, b=4, c=7, d=5
- d) a=1, b=2, c=3, d=4

**179. In the given C snippet, find the statement number that has error.**

```
//C code to push an element into a stack
1. void push( struct stack *s, int x)
2. {
3.     if(s->top==MAX-1)
4.     {
5.         printf("stack overflow");
6.     }
7.     else
8.     {
9.         s->items[++s->top]=x;
10.        s++;
11.    }
12. }
```

- a) 1
- b) 9
- c) 10**
- d) 11

**180. What is the other name for a postfix expression?**

- a) Normal polish Notation
- b) Reverse polish Notation**
- c) Warsaw notation
- d) Infix notation

**181. Which of the following is an example for a postfix expression?**

- a)  $a * b(c + d)$
- b)  $abc * + de - +$**
- c)  $+ab$
- d)  $a + b - c$

**182. Reverse Polish Notation is the reverse of a Polish Notation.**

- a) True
- b) False**

**183. What is the time complexity of evaluation of postfix expression algorithm?**

- a)  **$O(N)$**
- b)  $O(N \log N)$
- c)  $O(N^2)$
- d)  $O(M \log N)$

**184. In Postfix expressions, the operators come after the operands.**

- a) **True**
- b) False

**185. Which of these operators have the highest order of precedence?**

- a) '(' and ')'
- b) '\*' and '/'
- c) **'~' and '^'**
- d) '+' and '-'

**186. Which of the following is not an application of stack?**

- a) evaluation of postfix expression
- b) conversion of infix to postfix expression
- c) balancing symbols
- d) **line at ticket counter**

**187. While evaluating a postfix expression, when an operator is encountered, what is the correct operation to be performed?**

- a) push it directly on to the stack
- b) **pop 2 operands, evaluate them and push the result on to the stack**
- c) pop the entire stack
- d) ignore the operator

**188. Which of the following statement is incorrect?**

- a) **Postfix operators use value to their right**
- b) Postfix operators use value to their left
- c) Prefix operators use value to their right
- d) In postfix expression, operands are followed by operators

**189. What is the result of the given postfix expression?  $abc^{*+}$  where  $a=1$ ,  $b=2$ ,  $c=3$ .**

- a) 4
- b) 5
- c) 6
- d) **7**



**190. What is the result of the following postfix expression?**

**$ab*cd*+$**  where  $a=2, b=2, c=3, d=4$ .

- a) 16**
- b) 12
- c) 14
- d) 10

**191. Consider the stack**

| 5 |  
| 4 |  
| 3 |  
| 2 |.

**At this point, '\*' is encountered. What has to be done?**

- a)  $5*4=20$  is pushed into the stack**
- b) \* is pushed into the stack
- c)  $2*3=6$  is pushed into the stack
- d) \* is ignored

**192. Evaluate the postfix expression  $ab + cd/-$  where  $a=5, b=4, c=9, d=3$ .**

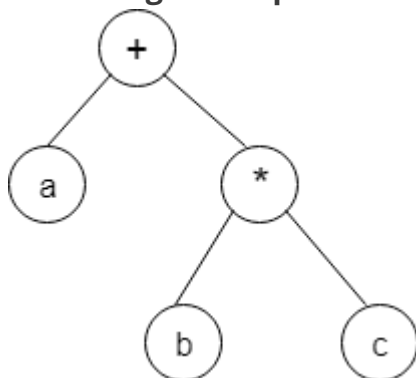
- a) 23
- b) 15
- c) 6**
- d) 10

**193. Evaluate and write the result for the following postfix expression**

**$abc*+de*f+g*+$**  where  $a=1, b=2, c=3, d=4, e=5, f=6, g=2$ .

- a) 61
- b) 59**
- c) 60
- d) 55

**194. For the given expression tree, write the correct postfix expression.**



- a)  $abc*+$**
- b)  $abc+*$
- c)  $ab+c*$
- d)  $a+bc*$

**195. What data structure is used when converting an infix notation to prefix notation?**

- a) **Stack**
- b) Queue
- c) B-Trees
- d) Linked-list

**196. What would be the Prefix notation for the given equation?**

$$A + (B * C)$$

- a)  $+A*CB$
- b)  $*B+AC$
- c)  **$+A*BC$**
- d)  $*A+CB$

**197. What would be the Prefix notation for the given equation?**

$$(A * B) + (C * D)$$

- a)  **$+*AB*CD$**
- b)  $*+AB*CD$
- c)  $**AB+CD$
- d)  $+*BA*CD$

**198. What would be the Prefix notation for the given equation?**

$$A + B * C ^ D$$

- a)  **$+A*B^CD$**
- b)  $+A^B*CD$
- c)  $*A+B^CD$
- d)  $^A*B+CD$

**199. Out of the following operators (^, \*, +, &, \$), the one having highest priority is**

- \_\_\_\_\_
- a) +
  - b) \$
  - c) **^**
  - d) &

**200. Out of the following operators (|, \*, +, &, \$), the one having lowest priority is**

- \_\_\_\_\_
- a) +
  - b) \$
  - c) **|**
  - d) &

**201. What would be the Prefix notation for the given equation?**

$A^B C^D$

**a)  $^{A B C}D$**

b)  $A^A B^A C^D$

c)  $A B C D^{A A A}$

d)  $A B^A C^A D$

**202. What would be the Prefix notation for the given equation?**

$a + b - c / d \& e | f$

**a)  $| \& - + a b / c d e f$**

b)  $\& | - + a b / c d e f$

c)  $| \& - a b + / c d e f$

d)  $| \& - + / a b c d e f$

**203. What would be the Prefix notation for the given equation?**

$(a + (b / c) * (d^e) - f)$

a)  $- + a * / ^ b c d e f$

**b)  $- + a * / b c ^ d e f$**

c)  $- + a * b / c ^ d e f$

d)  $- a + * / b c ^ d e f$

**204. What would be the Prefix notation and Postfix notation for the given equation?**

$A + B + C$

**a)  $++ABC$  and  $AB+C+$**

b)  $AB+C+$  and  $++ABC$

c)  $ABC++$  and  $AB+C+$

d)  $ABC+$  and  $ABC+$

**205. What would be the Prefix notation for the given equation?**

$a | b \& c$

a)  $a | \& b c$

b)  $\& | a b c$

**c)  $| a \& b c$**

d)  $a b \& | c$

**206. When an operand is read, which of the following is done?**

**a) It is placed on to the output**

b) It is placed in operator stack

c) It is ignored

d) Operator stack is emptied

**207. What should be done when a left parenthesis '(' is encountered?**

- a) It is ignored
- b) It is placed in the output
- c) It is placed in the operator stack**
- d) The contents of the operator stack is emptied

**208. Which of the following is an infix expression?**

- a) (a+b)\*(c+d)**
- b)  $ab+c^*$
- c)  $+ab$
- d)  $abc+^*$

**209. What is the time complexity of an infix to postfix conversion algorithm?**

- a)  $O(N \log N)$
- b)  $O(N)$**
- c)  $O(N^2)$
- d)  $O(M \log N)$

**210. What is the postfix expression for the corresponding infix expression?**

$a+b*c+(d*e)$

- a)  $abc^*+de^*+$**
- b)  $abc+^*de^*+$
- c)  $a+bc^*de+^*$
- d)  $abc^*+(de)^*+$

**211. Parentheses are simply ignored in the conversion of infix to postfix expression.**

- a) True
- b) False**

**212. It is easier for a computer to process a postfix expression than an infix expression.**

- a) True**
- b) False

**213. What is the postfix expression for the infix expression?**

$a-b-c$

- a)  $-ab-c$
- b)  $ab - c -$**
- c)  $- -abc$
- d)  $-ab-c$

**214. What is the postfix expression for the following infix expression?**

$a/b^c-d$

- a)  $abc^d/-$
- b)  $ab/cd^-$
- c)  $ab/^cd^-$
- d)  $abcd^/-$

**215. Which of the following statement is incorrect with respect to infix to postfix conversion algorithm?**

- a) operand is always placed in the output
- b) operator is placed in the stack when the stack operator has lower precedence
- c) parenthesis are included in the output**
- d) higher and equal priority operators follow the same condition

**216. In infix to postfix conversion algorithm, the operators are associated from?**

- a) right to left
- b) left to right**
- c) centre to left
- d) centre to right

**217. What is the corresponding postfix expression for the given infix expression?**

$a*(b+c)/d$

- a)  $ab^*+cd/$
- b)  $ab+^*cd/$
- c)  $abc^*+/d$
- d)  $abc+^*d/$**

**218. What is the corresponding postfix expression for the given infix expression?**

$a+(b*c(d/e^f)*g)*h$

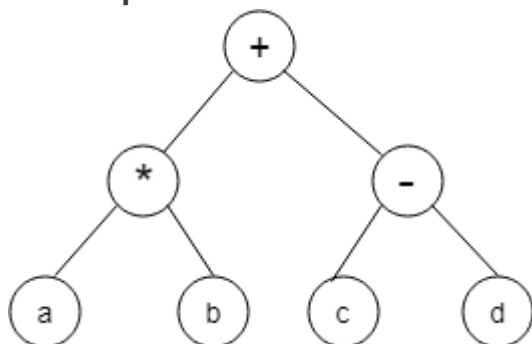
- a)  $ab^*cdef/^*g^*h+$
- b)  $abcdef/^*g^*h^*+$**
- c)  $abcd^*^ed/g^*-h^*+$
- d)  $abc^*de^fg/^*^*h+$

**219. What is the correct postfix expression for the following expression?**

$a+b*(c^d-e)^(f+g*h)-i$

- a)  $abc^de-fg+^*^*+i-$
- b)  $abcde^-fg^*+^*^*h^*+i-$
- c)  $abcd^e-fgh^*+^*^*+i-$**
- d)  $ab^-dc^*+ef^*gh^*+i-$

**220. From the given Expression tree, identify the correct postfix expression from the list of options.**



a)  $ab*cd*+$

**b)  $ab*cd-+$**

c)  $abcd-*+$

d)  $ab*+cd-$

**221. Given a prefix and a postfix notation what are the difference between them?**

**a) The postfix equation is solved starting from the left whereas the prefix notation is solved from the right**

b) The postfix equation is solved starting from the right whereas the prefix notation is solved from the left

c) Both equations are solved starting from the same side(right)

d) Both equations are solved starting from the same side(left)

**222. When converting the prefix notation into an infix notation, the first step to be followed is \_\_\_\_\_**

**a) Reverse the equation**

b) Push the equation to the stack

c) Push the equation onto the queue

d) Push the equation to the stack or queue

**223. The time complexity of converting a prefix notation to infix notation is \_\_\_\_\_**

**a)  $O(n)$  where  $n$  is the length of the equation**

b)  $O(n)$  where  $n$  is number of operands

c)  $O(1)$

d)  $O(\log n)$  where  $n$  is length of the equation

**224. Given two processes (conversion of postfix equation to infix notation and conversion of prefix notation to infix notation), which of the following is easier to implement?**

a) Both are easy to implement

b) Conversion of postfix equation to infix equation is harder than converting a prefix notation to infix notation

**c) Conversion of postfix equation to infix equation is easier than converting a prefix notation to infix notation**

d) Insufficient data

- 225. Which of the following data structure is used to convert postfix expression to infix expression?**
- a) **Stack**
  - b) Queue
  - c) Linked List
  - d) Heap
- 226. Consider the postfix expression 4 5 6 a b 7 8 a c, where a, b, c are operators. Operator a has higher precedence over operators b and c. Operators b and c are right associative. Then, equivalent infix expression is**
- a) 4 a 5 6 b 7 8 a c
  - b) 4 a 5 c 6 b 7 a 8
  - c) **4 b 5 a 6 c 7 a 8**
  - d) 4 a 5 b 6 c 7 a 8
- 227. To convert the postfix expression into the infix expression we use stack and scan the postfix expression from left to right.**
- a) **True**
  - b) False
- 228. Which of the following is valid reverse polish expression?**
- a) a op b
  - b) op a b
  - c) **a b op**
  - d) both op a b and a b op
- 229. How many children does a binary tree have?**
- a) 2
  - b) any number of children
  - c) **0 or 1 or 2**
  - d) 0 or 1
- 230. What is/are the disadvantages of implementing tree using normal arrays?**
- a) difficulty in knowing children nodes of a node
  - b) difficult in finding the parent of a node
  - c) **have to know the maximum number of nodes possible before creation of trees**
  - d) difficult to implement
- 231. What must be the ideal size of array if the height of tree is 'l'?**
- a)  **$2^l - 1$**
  - b)  **$l - 1$**
  - c) **l**
  - d)  **$2l$**

**232. What are the children for node 'w' of a complete-binary tree in an array representation?**

- a)  $2w$  and  $2w+1$
- b)  $2+w$  and  $2-w$
- c)  $w+1/2$  and  $w/2$
- d)  $w-1/2$  and  $w+1/2$

**233. What is the parent for a node 'w' of a complete binary tree in an array representation when w is not 0?**

- a)  $\text{floor}(w-1/2)$
- b)  $\text{ceil}(w-1/2)$
- c)  $w-1/2$
- d)  $w/2$

**234. If the tree is not a complete binary tree then what changes can be made for easy access of children of a node in the array?**

- a) every node stores data saying which of its children exist in the array
- b) no need of any changes continue with  $2w$  and  $2w+1$ , if node is at  $i$
- c) keep a separate table telling children of a node
- d) use another array parallel to the array with tree

**235. Consider a situation of writing a binary tree into a file with memory storage efficiency in mind, is array representation of tree is good?**

- a) yes because we are overcoming the need of pointers and so space efficiency
- b) yes because array values are indexable
- c) **No it is not efficient in case of sparse trees and remaining cases it is fine**
- d) No linked list representation of tree is only fine

**236. Can a tree stored in an array using either one of inorder or post order or pre order traversals be again reformed?**

- a) Yes just traverse through the array and form the tree
- b) **No we need one more traversal to form a tree**
- c) No in case of sparse trees
- d) Yes by using both inorder and array elements

**237. Advantages of linked list representation of binary trees over arrays?**

- a) dynamic size
- b) ease of insertion/deletion
- c) ease in randomly accessing a node
- d) **both dynamic size and ease in insertion/deletion**



**238. Disadvantages of linked list representation of binary trees over arrays?**

- a) Randomly accessing is not possible
- b) Extra memory for a pointer is needed with every element in the list
- c) Difficulty in deletion
- d) Random access is not possible and extra memory with every element**

**239. Which of the following traversing algorithm is not used to traverse in a tree?**

- a) Post order
- b) Pre order
- c) Post order
- d) Randomized**

**240. Level order traversal of a tree is formed with the help of**

- a) breadth first search**
- b) depth first search
- c) dijkstra's algorithm
- d) prims algorithm

**241. Identify the reason which doesn't play a key role to use threaded binary trees?**

- a) The storage required by stack and queue is more
- b) The pointers in most of nodes of a binary tree are NULL
- c) It is Difficult to find a successor node
- d) They occupy less size**

**242. The following lines talks about deleting a node in a binary tree.(the tree property must not be violated after deletion)**

**i) from root search for the node to be deleted**

**ii) \_\_\_\_\_**

**iii) delete the node at**

**what must be statement ii) and fill up statement iii)**

- a) ii)-find random node,replace with node to be deleted. iii)- delete the node
- b) ii)-find node to be deleted. iii)- delete the node at found location
- c) ii)-find deepest node,replace with node to be deleted. iii)- delete a node

**d) ii)-find deepest node,replace with node to be deleted. iii)- delete the deepest node**

**243. What may be the psuedo code for finding the size of a tree?**

- a)  $\text{find\_size}(\text{root\_node} \rightarrow \text{left\_node}) + 1 + \text{find\_size}(\text{root\_node} \rightarrow \text{right\_node})$**
- b)  $\text{find\_size}(\text{root\_node} \rightarrow \text{left\_node}) + \text{find\_size}(\text{root\_node} \rightarrow \text{right\_node})$
- c)  $\text{find\_size}(\text{root\_node} \rightarrow \text{right\_node}) - 1$
- d)  $\text{find\_size}(\text{root\_node} \rightarrow \text{left\_node}) + 1$

**244. What is missing in this logic of finding a path in the tree for a given sum (i.e checking whether there will be a path from roots to leaf nodes with given sum)?**

```
checkSum(struct bin-treenode *root , int sum) :
    if (root==null)
        return sum as 0
    else :
        leftover_sum=sum-root_node-->value
        //missing
```

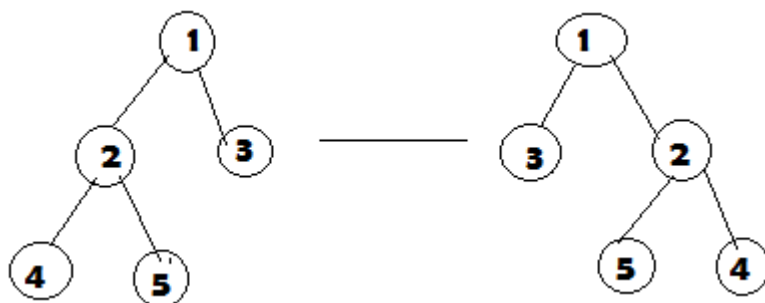
**a) code for having recursive calls to either only left tree or right trees or to both subtrees depending on their existence**

b) code for having recursive calls to either only left tree or right trees

c) code for having recursive calls to either only left tree

d) code for having recursive calls to either only right trees

**245. What must be the missing logic below so as to print mirror of a tree as below as an example?**



```
if (rootnode) :
    mirror(rootnode-->left)
    mirror(rootnode-->right)

    //missing

end
```

**a) swapping of left and right nodes is missing**

b) swapping of left with root nodes is missing

c) swapping of right with root nodes is missing

d) nothing is missing

**246. What is the code below trying to print?**

```
void print(tree *root, tree *node)
{
    if (root == null) return 0
    if (root-->left==node || root-->right==node) || print (root->left, node)
    || printf (root->right, node)
    {
        print (root->data)
    }
}
```

a) just printing all nodes

b) not a valid logic to do any task

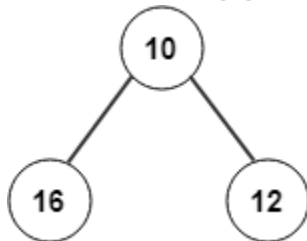
**c) printing ancestors of a node passed as argument**

d) printing nodes from leaf node to a node passed as argument

**247. What is the maximum number of children that a binary tree node can have?**

- a) 0
- b) 1
- c) 2**
- d) 3

**248. The following given tree is an example for?**



- a) Binary tree**
- b) Binary search tree
- c) Fibonacci tree
- d) AVL tree

**249. A binary tree is a rooted tree but not an ordered tree.**

- a) true
- b) false**

**250. How many common operations are performed in a binary tree?**

- a) 1
- b) 2
- c) 3**
- d) 4

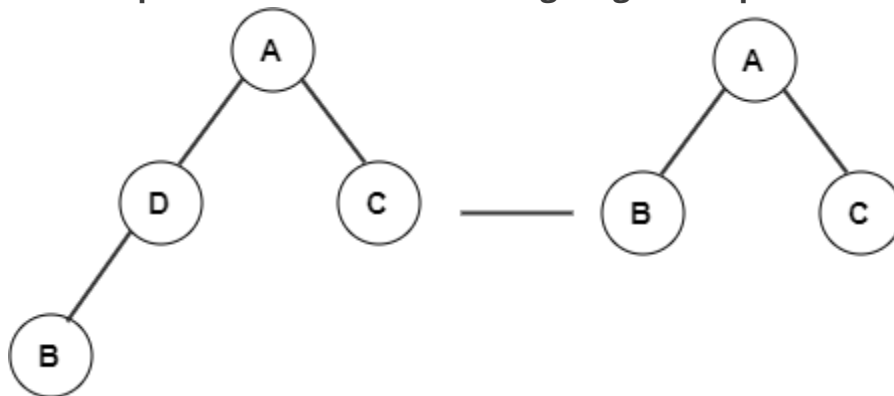
**251. What is the traversal strategy used in the binary tree?**

- a) depth-first traversal
- b) breadth-first traversal**
- c) random traversal
- d) Priority traversal

**252. How many types of insertion are performed in a binary tree?**

- a) 1
- b) 2**
- c) 3
- d) 4

253. What operation does the following diagram depict?



- a) inserting a leaf node
- b) inserting an internal node
- c) deleting a node with 0 or 1 child**
- d) deleting a node with 2 children

254. General ordered tree can be encoded into binary trees.

- a) true**
- b) false

255. How many bits would a succinct binary tree occupy?

- a)  $n + O(n)$
- b)  $2n + O(n)$**
- c)  $n/2$
- d)  $n$

256. The average depth of a binary tree is given as?

- a)  $O(N)$
- b)  $O(\sqrt{N})$
- c)  $O(N^2)$
- d)  $O(\log N)$**

257. How many orders of traversal are applicable to a binary tree (In General)?

- a) 1
- b) 4
- c) 2
- d) 3**

258. If binary trees are represented in arrays, what formula can be used to locate a left child, if the node has an index  $i$ ?

- a)  $2i+1$**
- b)  $2i+2$
- c)  $2i$
- d)  $4i$

**259. Using what formula can a parent node be located in an array?**

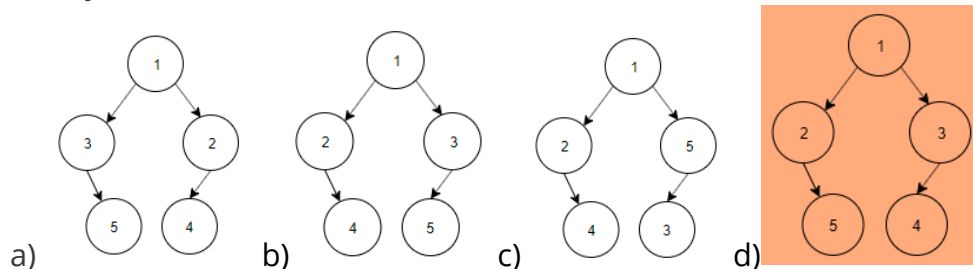
- a)  $(i+1)/2$
- b)  $(i-1)/2$**
- c)  $i/2$
- d)  $2i/2$

**260. Which of the following properties are obeyed by all three tree – traversals?**

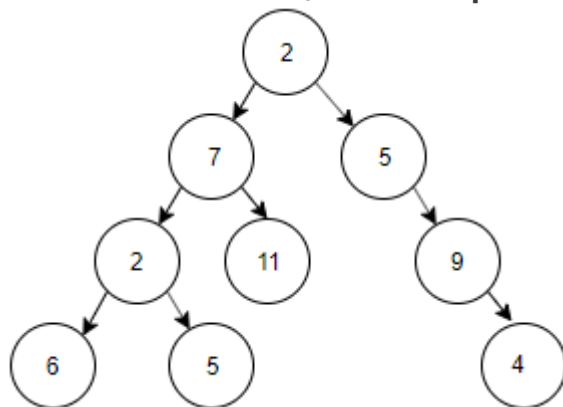
- a) Left subtrees are visited before right subtrees**
- b) Right subtrees are visited before left subtrees
- c) Root node is visited before left subtree
- d) Root node is visited before right subtree

**261. Construct a binary tree using the following data.**

The preorder traversal of a binary tree is 1, 2, 5, 3, 4. The inorder traversal of the same binary tree is 2, 5, 1, 4, 3.



**262. For the tree below, write the pre-order traversal.**



- a) 2, 7, 2, 6, 5, 11, 5, 9, 4**
- b) 2, 7, 5, 2, 6, 9, 5, 11, 4
- c) 2, 5, 11, 6, 7, 4, 9, 5, 2
- d) 2, 7, 5, 6, 11, 2, 5, 4, 9

**263. What is the time complexity of pre-order traversal in the iterative fashion?**

- a)  $O(1)$
- b)  $O(n)$**
- c)  $O(\log n)$
- d)  $O(n \log n)$

- 264. What is the space complexity of the post-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)**
- a)  $O(1)$
  - b)  $O(n \log d)$
  - c)  $O(\log d)$
  - d)  $O(d)$**
- 265. To obtain a prefix expression, which of the tree traversals is used?**
- a) Level-order traversal
  - b) Pre-order traversal**
  - c) Post-order traversal
  - d) In-order traversal
- 266. Consider the following data. The pre order traversal of a binary tree is A, B, E, C, D. The in order traversal of the same binary tree is B, E, A, D, C. The level order sequence for the binary tree is \_\_\_\_\_**
- a) A, C, D, B, E
  - b) A, B, C, D, E**
  - c) A, B, C, E, D
  - d) D, B, E, A, C
- 267. Consider the following data and specify which one is Preorder Traversal Sequence, Inorder and Postorder sequences.**
- S1: N, M, P, O, Q**
- S2: N, P, Q, O, M**
- S3: M, N, O, P, Q**
- a) S1 is preorder, S2 is inorder and S3 is postorder
  - b) S1 is inorder, S2 is preorder and S3 is postorder
  - c) S1 is inorder, S2 is postorder and S3 is preorder**
  - d) S1 is postorder, S2 is inorder and S3 is preorder
- 268. In postorder traversal of binary tree right subtree is traversed before visiting root.**
- a) True**
  - b) False
- 269. What is the possible number of binary trees that can be created with 3 nodes, giving the sequence N, M, L when traversed in post-order.**
- a) 15
  - b) 3
  - c) 5**
  - d) 8

- 270. The post-order traversal of a binary tree is O P Q R S T. Then possible pre-order traversal will be \_\_\_\_\_**
- a) T Q R S O P
  - b) T O Q R P S
  - c) T Q O P S R**
  - d) T Q O S P R
- 271. A binary search tree contains values 7, 8, 13, 26, 35, 40, 70, 75. Which one of the following is a valid post-order sequence of the tree provided the pre-order sequence as 35, 13, 7, 8, 26, 70, 40 and 75?**
- a) 7, 8, 26, 13, 75, 40, 70, 35
  - b) 26, 13, 7, 8, 70, 75, 40, 35
  - c) 7, 8, 13, 26, 35, 40, 70, 75
  - d) 8, 7, 26, 13, 40, 75, 70, 35**
- 272. Which of the following pair's traversals on a binary tree can build the tree uniquely?**
- a) post-order and pre-order
  - b) post-order and in-order**
  - c) post-order and level order
  - d) level order and preorder
- 273. A full binary tree can be generated using \_\_\_\_\_**
- a) post-order and pre-order traversal**
  - b) pre-order traversal
  - c) post-order traversal
  - d) in-order traversal
- 274. The maximum number of nodes in a tree for which post-order and pre-order traversals may be equal is \_\_\_\_\_**
- a) 3
  - b) 1**
  - c) 2
  - d) any number
- 275. The steps for finding post-order traversal are traverse the right subtree, traverse the left subtree or visit the current node.**
- a) True
  - b) False**

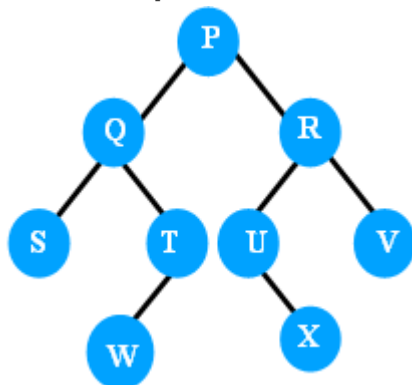
**276.** The pre-order and in-order are traversals of a binary tree are T M L N P O Q and L M N T O P Q. Which of following is post-order traversal of the tree?

- a) L N M O Q P T
- b) N M O P O L T
- c) L M N O P Q T
- d) O P L M N Q T

**277.** For a binary tree the first node visited in in-order and post-order traversal is same.

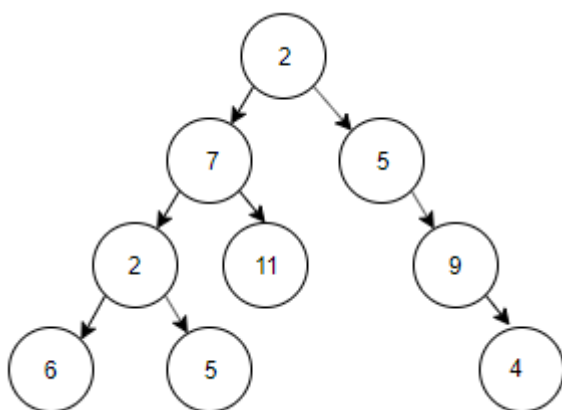
- a) True
- b) False

**278.** Find the postorder traversal of the binary tree shown below.



- a) P Q R S T U V W X
- b) W R S Q P V T U X
- c) S W T Q X U V R P
- d) S T W U X V Q R P

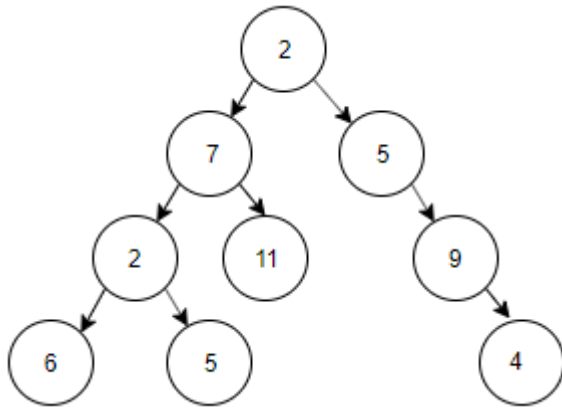
**279.** For the tree below, write the in-order traversal.



- a) 6, 2, 5, 7, 11, 2, 5, 9, 4
- b) 6, 5, 2, 11, 7, 4, 9, 5, 2
- c) 2, 7, 2, 6, 5, 11, 5, 9, 4
- d) 2, 7, 6, 5, 11, 2, 9, 5, 4



**280. For the tree below, write the level-order traversal.**



- a) 2, 7, 2, 6, 5, 11, 5, 9, 4
- b) 2, 7, 5, 2, 11, 9, 6, 5, 4**
- c) 2, 5, 11, 6, 7, 4, 9, 5, 2
- d) 2, 7, 5, 6, 11, 2, 5, 4, 9

**281. What is the space complexity of the in-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)**

- a)  $O(1)$
- b)  $O(n \log d)$
- c)  $O(\log d)$
- d)  $O(d)$**

**282. What is the time complexity of level order traversal?**

- a)  $O(1)$
- b)  $O(n)$**
- c)  $O(\log n)$
- d)  $O(n \log n)$

**283. Which of the following graph traversals closely imitates level order traversal of a binary tree?**

- a) Depth First Search
- b) Breadth First Search**
- c) Depth & Breadth First Search
- d) Binary Search

**284. In a binary search tree, which of the following traversals would print the numbers in the ascending order?**

- a) Level-order traversal
- b) Pre-order traversal
- c) Post-order traversal
- d) In-order traversal**

**285. The number of edges from the root to the node is called \_\_\_\_\_ of the tree.**

- a) Height
- b) Depth**
- c) Length
- d) Width

**286. The number of edges from the node to the deepest leaf is called \_\_\_\_\_ of the tree.**

- a) Height**
- b) Depth
- c) Length
- d) Width

**287. What is a full binary tree?**

- a) Each node has exactly zero or two children**
- b) Each node has exactly two children
- c) All the leaves are at the same level
- d) Each node has exactly one or two children

**288. What is a complete binary tree?**

- a) Each node has exactly zero or two children
- b) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from right to left
- c) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right**
- d) A tree in which all nodes have degree 2

**289. What is the average case time complexity for finding the height of the binary tree?**

- a)  $h = O(\log \log n)$
- b)  $h = O(n \log n)$
- c)  $h = O(n)$
- d)  $h = O(\log n)$**

**290. Which of the following is not an advantage of trees?**

- a) Hierarchical structure
- b) Faster search
- c) Router algorithms
- d) Undo/Redo operations in a notepad**

**291. In a full binary tree if number of internal nodes is I, then number of leaves L are?**

- a)  $L = 2 * I$
- b)  $L = I + 1$**
- c)  $L = I - 1$
- d)  $L = 2 * I - 1$

**292. In a full binary tree if number of internal nodes is I, then number of nodes N are?**

- a)  $N = 2 * I$
- b)  $N = I + 1$
- c)  $N = I - 1$
- d)  $N = 2 * I + 1$**

**293. In a full binary tree if there are L leaves, then total number of nodes N are?**

- a)  $N = 2 * L$
- b)  $N = L + 1$
- c)  $N = L - 1$
- d)  $N = 2 * L - 1$**

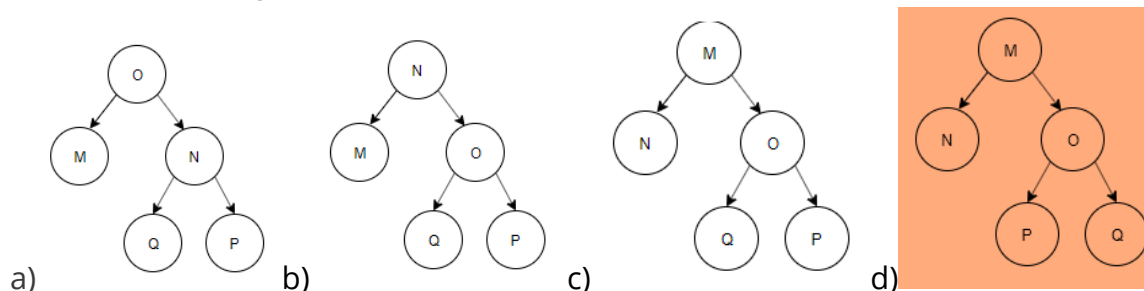
**294. Which of the following is incorrect with respect to binary trees?**

- a) Let T be a binary tree. For every  $k \geq 0$ , there are no more than  $2^k$  nodes in level k
- b) Let T be a binary tree with  $\lambda$  levels. Then T has no more than  $2^{\lambda-1}$  nodes
- c) Let T be a binary tree with N nodes. Then the number of levels is at least  $\text{ceil}(\log(N + 1))$
- d) Let T be a binary tree with N nodes. Then the number of levels is at least  $\text{floor}(\log(N + 1))$**

**295. Construct a binary tree by using postorder and inorder sequences given below.**

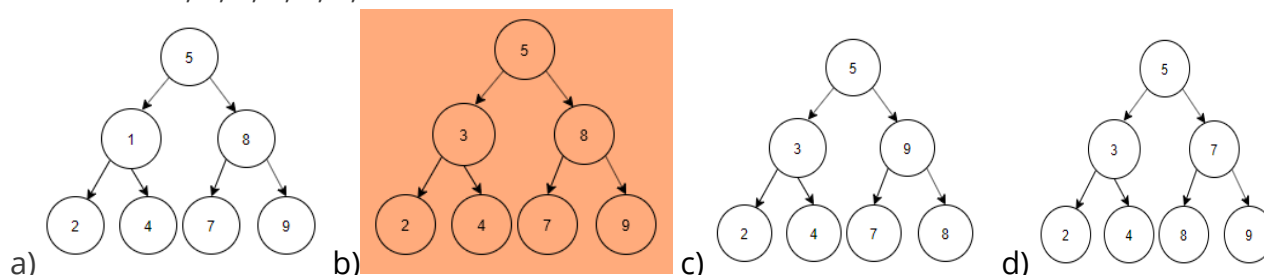
Inorder: N, M, P, O, Q

Postorder: N, P, Q, O, M



**296. Construct a binary search tree by using postorder sequence given below.**

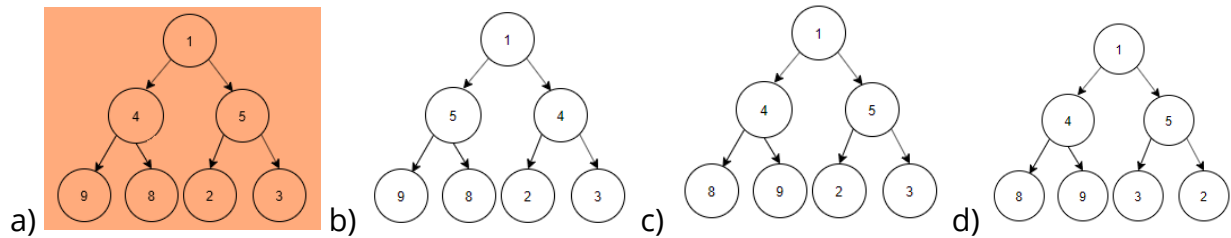
Postorder: 2, 4, 3, 7, 9, 8, 5.



**297.** Construct a binary tree using inorder and level order traversal given below.

Inorder Traversal: 3, 4, 2, 1, 5, 8, 9

Level Order Traversal: 1, 4, 5, 9, 8, 2, 3



**298.** Which of the following is false about a binary search tree?

- a) The left child is always lesser than its parent
- b) The right child is always greater than its parent
- c) The left and right sub-trees should also be binary search trees
- d) In order sequence gives decreasing order of elements**

**299.** What is the speciality about the inorder traversal of a binary search tree?

- a) It traverses in a non increasing order
- b) It traverses in an increasing order**
- c) It traverses in a random fashion
- d) It traverses based on priority of the node

**300.** What are the worst case and average case complexities of a binary search tree?

- a)  $O(n)$ ,  $O(n)$
- b)  $O(\log n)$ ,  $O(\log n)$
- c)  $O(\log n)$ ,  $O(n)$
- d)  $O(n)$ ,  $O(\log n)$**

**301.** What are the conditions for an optimal binary search tree and what is its advantage?

- a) The tree should not be modified and you should know how often the keys are accessed, it improves the lookup cost**
- b) You should know the frequency of access of the keys, improves the lookup time
- c) The tree can be modified and you should know the number of elements in the tree before hand, it improves the deletion time
- d) The tree should be just modified and improves the lookup time

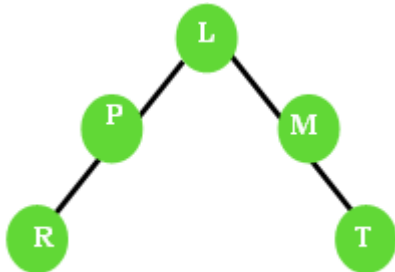
**302.** What will be the height of a balanced full binary tree with 8 leaves?

- a) 8
- b) 5
- c) 6
- d) 4**

**303. The balance factor of a node in a binary tree is defined as \_\_\_\_**

- a) addition of heights of left and right subtrees
- b) height of right subtree minus height of left subtree
- c) height of left subtree minus height of right subtree**
- d) height of right subtree minus one

**304. Figure below is a balanced binary tree. If a node inserted as child of the node R, how many nodes will become unbalanced?**



- a) 2
- b) 1**
- c) 3
- d) 0

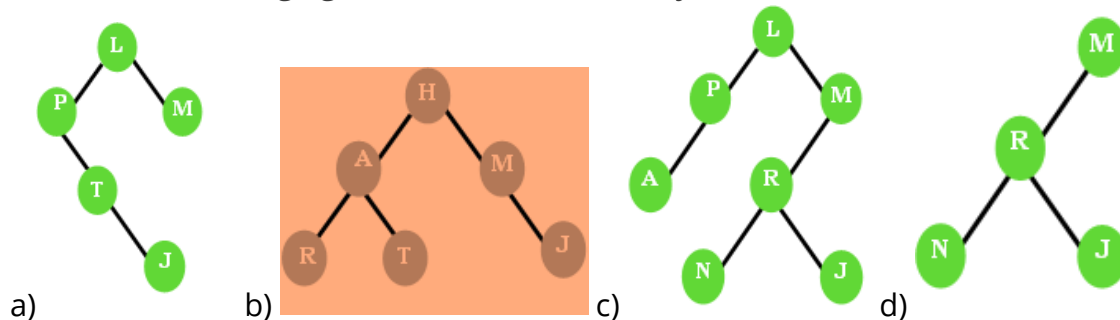
**305. A binary tree is balanced if the difference between left and right subtree of every node is not more than \_\_\_\_**

- a) 1**
- b) 3
- c) 2
- d) 0

**306. Which of the following tree data structures is not a balanced binary tree?**

- a) AVL tree
- b) Red-black tree
- c) Splay tree
- d) B-tree**

**307. Which of following figures is a balanced binary tree?**



**308. Balanced binary tree with  $n$  items allows the lookup of an item in \_\_\_\_ worst-case time.**

- a)  $O(\log n)$
- b)  $O(n \log 2)$
- c)  $O(n)$
- d)  $O(1)$

**309. Which of the following data structures can be efficiently implemented using height balanced binary search tree?**

- a) sets
- b) priority queue
- c) heap
- d) both sets and priority queue

**310. Two balanced binary trees are given with  $m$  and  $n$  elements respectively. They can be merged into a balanced binary search tree in \_\_\_\_ time.**

- a)  $O(m+n)$
- b)  $O(mn)$
- c)  $O(m)$
- d)  $O(m \log n)$

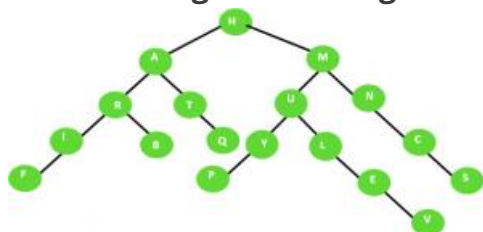
**311. Which of the following is an advantage of balanced binary search tree, like AVL tree, compared to binary heap?**

- a) insertion takes less time
- b) deletion takes less time
- c) searching takes less time
- d) construction of the tree takes less time than binary heap

**312. AVL trees are more balanced than Red-black trees.**

- a) True
- b) False

**313. The figure shown below is a balanced binary tree. If node P is deleted, which of the following nodes will get unbalanced?**



- a) U
- b) M
- c) H
- d) A

**314. Which of the following is not the self balancing binary search tree?**

- a) AVL Tree
- b) 2-3-4 Tree**
- c) Red – Black Tree
- d) Splay Tree

**315. The binary tree sort implemented using a self – balancing binary search tree takes \_\_\_\_\_ time is worst case.**

- a)  $O(n \log n)$**
- b)  $O(n)$
- c)  $O(n^2)$
- d)  $O(\log n)$

**316. An AVL tree is a self – balancing binary search tree, in which the heights of the two child sub trees of any node differ by \_\_\_\_\_**

- a) At least one
- b) At most one**
- c) Two
- d) At most two

**317. Associative arrays can be implemented using \_\_\_\_\_**

- a) B-tree
- b) A doubly linked list
- c) A single linked list
- d) A self balancing binary search tree**

**318. Self – balancing binary search trees have a much better average-case time complexity than hash tables.**

- a) True
- b) False**

**319. Which of the following is a self – balancing binary search tree?**

- a) 2-3 tree
- b) Threaded binary tree
- c) AA tree**
- d) Treap

**320. A self – balancing binary search tree can be used to implement \_\_\_\_\_**

- a) Priority queue**
- b) Hash table
- c) Heap sort
- d) Priority queue and Heap sort

- 321. In which of the following self – balancing binary search tree the recently accessed element can be accessed quickly?**
- a) AVL tree
  - b) AA tree
  - c) Splay tree**
  - d) Red – Black tree
- 322. The minimum height of self balancing binary search tree with n nodes is \_\_\_\_**
- a)  $\log_2(n)$**
  - b) n
  - c)  $2n + 1$
  - d)  $2n - 1$
- 323. Binary tree sort implemented using a self balancing binary search tree takes  $O(n \log n)$  time in the worst case but still it is slower than merge sort.**
- a) True**
  - b) False
- 324. What is a hash table?**
- a) A structure that maps values to keys
  - b) A structure that maps keys to values**
  - c) A structure used for storage
  - d) A structure used to implement stack and queue
- 325. If several elements are competing for the same bucket in the hash table, what is it called?**
- a) Diffusion
  - b) Replication
  - c) Collision**
  - d) Duplication
- 326. What is direct addressing?**
- a) Distinct array position for every possible key**
  - b) Fewer array positions than keys
  - c) Fewer keys than array positions
  - d) Same array position for all keys
- 327. What is the search complexity in direct addressing?**
- a)  $O(n)$
  - b)  $O(\log n)$
  - c)  $O(n \log n)$
  - d)  $O(1)$**



**328. What is a hash function?**

- a) A function has allocated memory to keys
- b) A function that computes the location of the key in the array**
- c) A function that creates an array
- d) A function that computes the location of the values in the array

**329. Which of the following is not a technique to avoid a collision?**

- a) Make the hash function appear random
- b) Use the chaining method
- c) Use uniform hashing
- d) Increasing hash table size**

**330. What is the load factor?**

- a) Average array size
- b) Average key size
- c) Average chain length**
- d) Average hash table length

**331. What is simple uniform hashing?**

- a) Every element has equal probability of hashing into any of the slots**
- b) A weighted probabilistic method is used to hash elements into the slots
- c) Elements has Random probability of hashing into array slots
- d) Elements are hashed based on priority

**332. In simple uniform hashing, what is the search complexity?**

- a)  $O(n)$
- b)  $O(\log n)$
- c)  $O(n \log n)$
- d)  $O(1)$**

**333. In simple chaining, what data structure is appropriate?**

- a) Singly linked list
- b) Doubly linked list**
- c) Circular linked list
- d) Binary trees

**334. Where is linear searching used?**

- a) When the list has only a few elements
- b) When performing a single search in an unordered list
- c) Used all the time
- d) When the list has only a few elements and When performing a single search in an unordered list**

**335.** Select the code snippet which performs unordered linear search iteratively?

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            break;
        }
    }
    return index;
}
```

(A)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == data)
        {
            break;
        }
    }
    return index;
}
```

(B)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i <= size; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            break;
        }
    }
    return index;
}
```

(C)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size-1; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            break;
        }
    }
    return index;
}
```

(D)

**336. What is the best case for linear search?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(1)$**

**337. What is the worst case for linear search?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$**
- d)  $O(1)$

**338. What is the best case and worst case complexity of ordered linear search?**

- a)  $O(n \log n)$ ,  $O(\log n)$
- b)  $O(\log n)$ ,  $O(n \log n)$
- c)  $O(n)$ ,  $O(1)$
- d)  $O(1)$ ,  $O(n)$**

**339. Which of the following is a disadvantage of linear search?**

- a) Requires more space
- b) Greater time complexities compared to other searching algorithms**
- c) Not easy to understand
- d) Not easy to implement

**340. Is there any difference in the speed of execution between linear search(recursive) vs linear search(iterative)?**

- a) Both execute at same speed
- b) Linear search(recursive) is faster
- c) Linear search(iterative) is faster**
- d) Can't be said

**341. Is the space consumed by the linear search(recursive) and linear search(iterative) same?**

- a) No, recursive algorithm consumes more space**
- b) No, recursive algorithm consumes less space
- c) Yes
- d) Nothing can be said

**342. What is the worst case runtime of linear search(recursive) algorithm?**

- a)  $O(n)$**
- b)  $O(\log n)$
- c)  $O(n^2)$
- d)  $O(n \times)$

**343. Linear search(recursive) algorithm used in \_\_\_\_\_**

- a) When the size of the dataset is low**
- b) When the size of the dataset is large
- c) When the dataset is unordered
- d) Never used

**344. The array is as follows: 1,2,3,6,8,10. At what time the element 6 is found? (By using linear search(recursive) algorithm)**

- a) 4th call**
- b) 3rd call
- c) 6th call
- d) 5th call

**345. The array is as follows: 1,2,3,6,8,10. Given that the number 17 is to be searched. At which call it tells that there's no such element? (By using linear search(recursive) algorithm)**

**a) 7th call**

b) 9th call

c) 17th call

d) The function calls itself infinite number of times

**346. What is the best case runtime of linear search(recursive) algorithm on an ordered set of elements?**

**a)  $O(1)$**

b)  $O(n)$

c)  $O(\log n)$

d)  $O(nx)$

**347. Which of the following code snippet performs linear search recursively?**

```
for(i=0;i<n;i++)
{
    if(a[i]==key)
        printf("element found");
}
```

(A)

```
LinearSearch(int[] a, n, key)
{
    if(n<1)
        return False
    if(a[n]==key)
        return True
    else
        LinearSearch(a, n-1, key)
}
```

(B)

```
LinearSearch(int[] a, n, key)
{
    if(n<1)
        return True
    if(a[n]==key)
        return False
    else
        LinearSearch(a, n-1, key)
}
```

(C)

```
LinearSearch(int[] a, n, key)
{
    if(n<1)
        return False
    if(a[n]==key)
        return True
    else
        LinearSearch(a, n+1, key)
}
```

(D)

**348. Can linear search recursive algorithm and binary search recursive algorithm be performed on an unordered list?**

**a) Binary search can't be used**

b) Linear search can't be used

c) Both cannot be used

d) Both can be used

**349. What is the recurrence relation for the linear search recursive algorithm?**

a)  $T(n-2)+c$

b)  $2T(n-1)+c$

**c)  $T(n-1)+c$**

d)  $T(n+1)+c$

**350. What is the advantage of recursive approach than an iterative approach?**

- a) Consumes less memory
- b) Less code and easy to implement**
- c) Consumes more memory
- d) More code has to be written

**351. Given an input arr = {2,5,7,99,899}; key = 899; What is the level of recursion for Binary Search?**

- a) 5
- b) 2
- c) 3**
- d) 4

**352. Given an array arr = {45,77,89,90,94,99,100} and key = 99; what are the mid values(corresponding array elements) in the first and second levels of recursion?**

- a) 90 and 99**
- b) 90 and 94
- c) 89 and 99
- d) 89 and 94

**353. What is the worst case complexity of binary search using recursion?**

- a)  $O(n \log n)$
- b)  $O(\log n)$**
- c)  $O(n)$
- d)  $O(n^2)$

**354. What is the average case time complexity of binary search using recursion?**

- a)  $O(n \log n)$
- b)  $O(\log n)$**
- c)  $O(n)$
- d)  $O(n^2)$

**355. Which of the following is not an application of binary search?**

- a) To find the lower/upper bound in an ordered sequence
- b) Union of intervals
- c) Debugging
- d) To search in unordered list**

**356. Binary Search can be categorized into which of the following?**

- a) Brute Force technique
- b) Divide and conquer**
- c) Greedy algorithm
- d) Dynamic programming

**357. Given an array arr = {5,6,77,88,99} and key = 88; How many iterations are done until the element is found?**

- a) 1
- b) 3
- c) 4
- d) 2**

**358. Given an array arr = {45,77,89,90,94,99,100} and key = 100; What are the mid values(corresponding array elements) generated in the first and second iterations?**

- a) 90 and 99**
- b) 90 and 100
- c) 89 and 94
- d) 94 and 99

**359. What is the time complexity of binary search with iteration?**

- a)  $O(n \log n)$
- b)  $O(\log n)$**
- c)  $O(n)$
- d)  $O(n^2)$

**360. How many passes does an insertion sort algorithm consist of?**

- a) N
- b) N-1**
- c) N+1
- d)  $N^2$

**361. Which of the following algorithm implementations is similar to that of an insertion sort?**

- a) Binary heap**
- b) Quick sort
- c) Merge sort
- d) Radix sort

**362. What is the average case running time of an insertion sort algorithm?**

- a)  $O(N)$
- b)  $O(N \log N)$
- c)  $O(\log N)$
- d)  $O(N^2)$**

**363. Any algorithm that sorts by exchanging adjacent elements require  $O(N^2)$  on average.**

- a) True**
- b) False

**364. What is the average number of inversions in an array of N distinct numbers?**

- a)  $N(N-1)/4$
- b)  $N(N+1)/2$
- c)  $N(N-1)/2$
- d)  $N(N-1)/3$

**365. What is the running time of an insertion sort algorithm if the input is pre-sorted?**

- a)  $O(N^2)$
- b)  $O(N \log N)$
- c)  **$O(N)$**
- d)  $O(M \log N)$

**366. What will be the number of passes to sort the elements using insertion sort?**

**14, 12, 16, 6, 3, 10**

- a) 6
- b) **5**
- c) 7
- d) 1

**367. For the following question, how will the array elements look like after second pass using insertion sort?**

**34, 8, 64, 51, 32, 21**

- a) 8, 21, 32, 34, 51, 64
- b) 8, 32, 34, 51, 64, 21
- c) 8, 34, 51, 64, 32, 21
- d) **8, 34, 64, 51, 32, 21**

**368. Which of the following real time examples is based on insertion sort?**

- a) **arranging a pack of playing cards**
- b) database scenarios and distributes scenarios
- c) arranging books on a library shelf
- d) real-time systems

**369. In C, what are the basic loops required to perform an insertion sort?**

- a) do- while
- b) if else
- c) **for and while**
- d) for and if

**370. Binary search can be used in an insertion sort algorithm to reduce the number of comparisons.**

- a) **True**
- b) False

**371. Which of the following options contain the correct feature of an insertion sort algorithm?**

- a) anti-adaptive
- b) dependable
- c) stable, not in-place
- d) stable, adaptive**

**372. Which of the following sorting algorithms is the fastest for sorting small arrays?**

- a) Quick sort
- b) Insertion sort**
- c) Shell sort
- d) Heap sort

**373. For the best case input, the running time of an insertion sort algorithm is?**

- a) Linear**
- b) Binary
- c) Quadratic
- d) Depends on the input

**374. Which of the following examples represent the worst case input for an insertion sort?**

- a) array in sorted order
- b) array sorted in reverse order**
- c) normal unsorted array
- d) large array

**375. Which of the following is correct with regard to insertion sort?**

- a) insertion sort is stable and it sorts In-place**
- b) insertion sort is unstable and it sorts In-place
- c) insertion sort is stable and it does not sort In-place
- d) insertion sort is unstable and it does not sort In-place

**376. Which of the following sorting algorithm is best suited if the elements are already sorted?**

- a) Heap Sort
- b) Quick Sort
- c) Insertion Sort**
- d) Merge Sort



**377. The worst case time complexity of insertion sort is  $O(n^2)$ . What will be the worst case time complexity of insertion sort if the correct position for inserting element is calculated using binary search?**

- a)  $O(n \log n)$
- b)  $O(n^2)$**
- c)  $O(n)$
- d)  $O(\log n)$

**378. Insertion sort is an example of an incremental algorithm.**

- a) True**
- b) False

**379. Consider the code given below, which runs insertion sort:**

```
void insertionSort(int arr[], int array_size)
{
    int i, j, value;
    for (i = 1; i < array_size; i++)
    {
        value = arr[i];
        j = i;
        while (_____ )
        {
            arr[j] = arr[j - 1];
            j = j - 1;
        }
        arr[j] = value;
    }
}
```

**Which condition will correctly implement the while loop?**

- a)  $(j > 0) \mid \mid (arr[j - 1] > value)$
- b)  $(j > 0) \&\& (arr[j - 1] > value)$**
- c)  $(j > 0) \&\& (arr[j + 1] > value)$
- d)  $(j > 0) \&\& (arr[j + 1] < value)$

**380. Which of the following is good for sorting arrays having less than 100 elements?**

- a) Quick Sort
- b) Selection Sort
- c) Merge Sort
- d) Insertion Sort**

**381. Consider an array of length 5,  $arr[5] = \{9, 7, 4, 2, 1\}$ . What are the steps of insertions done while running insertion sort on the array?**

- a) 7 9 4 2 1    4 7 9 2 1    2 4 7 9 1    1 2 4 7 9**
- b) 9 7 4 1 2    9 7 1 2 4    9 1 2 4 7    1 2 4 7 9
- c) 7 4 2 1 9    4 2 1 9 7    2 1 9 7 4    1 9 7 4 2
- d) 7 9 4 2 1    2 4 7 9 1    4 7 9 2 1    1 2 4 7 9

**382. Statement 1: In insertion sort, after m passes through the array, the first m elements are in sorted order.**

**Statement 2: And these elements are the m smallest elements in the array.**

a) Both the statements are true

**b) Statement 1 is true but statement 2 is false**

c) Statement 1 is false but statement 2 is true

d) Both the statements are false

**383. In insertion sort, the average number of comparisons required to place the 7<sup>th</sup> element into its correct position is \_\_\_\_**

a) 9

**b) 4**

c) 7

d) 14

**384. Which of the following is not an exchange sort?**

a) Bubble Sort

b) Quick Sort

c) Partition-exchange Sort

**d) Insertion Sort**

**385. What is an in-place sorting algorithm?**

**a) It needs  $O(1)$  or  $O(\log n)$  memory to create auxiliary locations**

b) The input is already sorted and in-place

c) It requires additional storage

d) It requires additional space

**386. In the following scenarios, when will you use selection sort?**

a) The input is already sorted

b) A large file has to be sorted

**c) Large values need to be sorted with small keys**

d) Small values need to be sorted with large keys

**387. What is the worst case complexity of selection sort?**

a)  $O(n \log n)$

b)  $O(\log n)$

c)  $O(n)$

**d)  $O(n^2)$**

**388. What is the advantage of selection sort over other sorting techniques?**

**a) It requires no additional storage space**

b) It is scalable

c) It works best for inputs which are already sorted

d) It is faster than any other sorting technique

**389. What is the average case complexity of selection sort?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$**

**390. What is the disadvantage of selection sort?**

- a) It requires auxiliary memory
- b) It is not scalable**
- c) It can be used for small keys
- d) It takes linear time to sort the elements

**391. The given array is  $arr = \{3,4,5,2,1\}$ . The number of iterations in bubble sort and selection sort respectively are \_\_\_\_\_**

- a) 5 and 4**
- b) 4 and 5
- c) 2 and 4
- d) 2 and 5

**392. The given array is  $arr = \{1,2,3,4,5\}$ . (bubble sort is implemented with a flag variable)The number of iterations in selection sort and bubble sort respectively are \_\_\_\_\_**

- a) 5 and 4
- b) 1 and 4
- c) 0 and 4
- d) 4 and 1**

**393. What is the best case complexity of selection sort?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$**

**394. What is an external sorting algorithm?**

- a) Algorithm that uses tape or disk during the sort**
- b) Algorithm that uses main memory during the sort
- c) Algorithm that involves swapping
- d) Algorithm that are considered 'in place'

**395. What is an internal sorting algorithm?**

- a) Algorithm that uses tape or disk during the sort
- b) Algorithm that uses main memory during the sort**
- c) Algorithm that involves swapping
- d) Algorithm that are considered 'in place'

**396. What is the worst case complexity of bubble sort?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$**

**397. What is the average case complexity of bubble sort?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$**

**398. Which of the following is not an advantage of optimised bubble sort over other sorting techniques in case of sorted elements?**

- a) It is faster
- b) Consumes less memory
- c) Detects whether the input is already sorted**
- d) Consumes less time

**399. The given array is  $arr = \{1, 2, 4, 3\}$ . Bubble sort is used to sort the array elements. How many iterations will be done to sort the array?**

- a) 4**
- b) 2
- c) 1
- d) 0

**400. What is the best case efficiency of bubble sort in the improvised version?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$**
- d)  $O(n^2)$

**401. The given array is  $arr = \{1, 2, 4, 3\}$ . Bubble sort is used to sort the array elements. How many iterations will be done to sort the array with improvised version?**

- a) 4
- b) 2**
- c) 1
- d) 0

**402. Merge sort uses which of the following technique to implement sorting?**

- a) backtracking
- b) greedy algorithm
- c) divide and conquer**
- d) dynamic programming

**403. What is the average case time complexity of merge sort?**

- a)  $O(n \log n)$**
- b)  $O(n^2)$
- c)  $O(n^2 \log n)$
- d)  $O(n \log n^2)$

**404. What is the auxiliary space complexity of merge sort?**

- a)  $O(1)$
- b)  $O(\log n)$
- c)  $O(n)$**
- d)  $O(n \log n)$

**405. Merge sort can be implemented using  $O(1)$  auxiliary space.**

- a) true**
- b) false

**406. What is the worst case time complexity of merge sort?**

- a)  $O(n \log n)$**
- b)  $O(n^2)$
- c)  $O(n^2 \log n)$
- d)  $O(n \log n^2)$

**407. Which of the following method is used for sorting in merge sort?**

- a) merging**
- b) partitioning
- c) selection
- d) exchanging

**408. What will be the best case time complexity of merge sort?**

- a)  $O(n \log n)$**
- b)  $O(n^2)$
- c)  $O(n^2 \log n)$
- d)  $O(n \log n^2)$

**409. Which of the following is not a variant of merge sort?**

- a) in-place merge sort
- b) bottom up merge sort
- c) top down merge sort
- d) linear merge sort**

**410. Choose the incorrect statement about merge sort from the following?**

- a) it is a comparison based sort
- b) it is an adaptive algorithm**
- c) it is not an in place algorithm
- d) it is stable algorithm

**411. Which of the following is not in place sorting algorithm by default?**

- a) merge sort**
- b) quick sort
- c) heap sort
- d) insertion sort

**412. Which of the following is not a stable sorting algorithm?**

- a) Quick sort**
- b) Cocktail sort
- c) Bubble sort
- d) Merge sort

**413. Which of the following stable sorting algorithm takes the least time when applied to an almost sorted array?**

- a) Quick sort
- b) Insertion sort
- c) Selection sort
- d) Merge sort**

**414. Merge sort is preferred for arrays over linked lists.**

- a) true
- b) false**

**415. Which of the following sorting algorithm makes use of merge sort?**

- a) tim sort**
- b) intro sort
- c) bogo sort
- d) quick sort

**416. Which of the following sorting algorithm does not use recursion?**

- a) quick sort
- b) merge sort
- c) heap sort
- d) bottom up merge sort**

**417. Which of the following sorting algorithms is the fastest?**

- a) Merge sort
- b) Quick sort**
- c) Insertion sort
- d) Shell sort

**418. Quick sort follows Divide-and-Conquer strategy.**

- a) True**
- b) False

**419. What is the worst case time complexity of a quick sort algorithm?**

- a)  $O(N)$
- b)  $O(N \log N)$
- c)  $O(N^2)$**
- d)  $O(\log N)$

**420. Which of the following methods is the most effective for picking the pivot element?**

- a) first element
- b) last element
- c) median-of-three partitioning**
- d) random element

**421. Find the pivot element from the given input using median-of-three partitioning method.**

**8, 1, 4, 9, 6, 3, 5, 2, 7, 0.**

- a) 8
- b) 7
- c) 9
- d) 6**

**422. Which is the safest method to choose a pivot element?**

- a) choosing a random element as pivot**
- b) choosing the first element as pivot
- c) choosing the last element as pivot
- d) median-of-three partitioning method

**423. What is the average running time of a quick sort algorithm?**

- a)  $O(N^2)$
- b)  $O(N)$
- c)  $O(N \log N)$**
- d)  $O(\log N)$

**424. Which of the following sorting algorithms is used along with quick sort to sort the sub arrays?**

- a) Merge sort
- b) Shell sort
- c) Insertion sort**
- d) Bubble sort

**425. Quick sort uses join operation rather than merge operation.**

- a) true**
- b) false

**426. How many sub arrays does the quick sort algorithm divide the entire array into?**

- a) one
- b) two**
- c) three
- d) four

**427. Which is the worst method of choosing a pivot element?**

- a) first element as pivot**
- b) last element as pivot
- c) median-of-three partitioning
- d) random element as pivot

**428. Which among the following is the best cut-off range to perform insertion sort within a quick sort?**

- a)  $N=0-5$
- b)  $N=5-20$**
- c)  $N=20-30$
- d)  $N>30$

**429. Quick sort is a \_\_\_\_\_**

- a) greedy algorithm
- b) divide and conquer algorithm**
- c) dynamic programming algorithm
- d) backtracking algorithm

**430. What is the worst case time complexity of the Quick sort?**

- a)  $O(n \log n)$
- b)  $O(n)$
- c)  $O(n^3)$
- d)  $O(n^2)$**

**431. Apply Quick sort on a given sequence 7 11 14 6 9 4 3 12. What is the sequence after first phase, pivot is first element?**

- a) 6 4 3 7 11 9 14 12
- b) 6 3 4 7 9 14 11 12**
- c) 7 6 14 11 9 4 3 12
- d) 7 6 4 3 9 14 11 12

**432. The best case behaviour occurs for quick sort is, if partition splits the array of size  $n$  into \_\_\_\_\_**

- a)  $n/2 : (n/2) - 1$**
- b)  $n/2 : n/3$
- c)  $n/4 : 3n/2$
- d)  $n/4 : 3n/4$



**433. Quick sort is a stable sorting algorithm.**

- a) True
- b) False**

**434. Consider the Quick sort algorithm in which the partitioning procedure splits elements into two sub-arrays and each sub-array contains at least one-fourth of the elements. Let  $T(n)$  be the number of comparisons required to sort array of  $n$  elements. Then  $T(n) \leq$ ?**

- a)  $T(n) \leq 2 T(n/4) + cn$
- b)  $T(n) \leq T(n/4) + T(3n/4) + cn$**
- c)  $T(n) \leq 2 T(3n/4) + cn$
- d)  $T(n) \leq T(n/3) + T(3n/4) + cn$

**435. Consider the Quick sort algorithm which sorts elements in ascending order using the first element as pivot. Then which of the following input sequence will require a maximum number of comparisons when this algorithm is applied on it?**

- a) 22 25 56 67 89**
- b) 52 25 76 67 89
- c) 22 25 76 67 50
- d) 52 25 89 67 76

**436. A machine needs a minimum of 200 sec to sort 1000 elements by Quick sort. The minimum time needed to sort 200 elements will be approximately \_\_\_\_\_**

- a) 60.2 sec
- b) 45.54 sec
- c) 31.11 sec**
- d) 20 sec

**437. Which one of the following sorting algorithm is best suited to sort an array of 1 million elements?**

- a) Bubble sort
- b) Insertion sort
- c) Merge sort
- d) Quick sort**

**438. Quick sort is a space-optimised version of \_\_\_\_**

- a) Bubble sort
- b) Selection sort
- c) Insertion sort
- d) Binary tree sort**

**439. QuickSort can be categorized into which of the following?**

- a) Brute Force technique
- b) Divide and conquer**
- c) Greedy algorithm
- d) Dynamic programming

**440. What is the worst case complexity of QuickSort?**

- a)  $O(n \log n)$
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$**

**441. What is a randomized QuickSort?**

- a) The leftmost element is chosen as the pivot
- b) The rightmost element is chosen as the pivot
- c) Any element in the array is chosen as the pivot**
- d) A random number is generated which is used as the pivot

**442. What is the best case complexity of QuickSort?**

- a)  $O(n \log n)$**
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$

**443. The given array is arr = {2,3,4,1,6}. What are the pivots that are returned as a result of subsequent partitioning?**

- a) 1 and 3**
- b) 3 and 1
- c) 2 and 6
- d) 6 and 2

**444. What is the average case complexity of QuickSort?**

- a)  $O(n \log n)$**
- b)  $O(\log n)$
- c)  $O(n)$
- d)  $O(n^2)$

**445. The given array is arr = {2,6,1}. What are the pivots that are returned as a result of subsequent partitioning?**

- a) 1 and 6
- b) 6 and 1
- c) 2 and 6
- d) 1**

**446. Which of the following is not true about QuickSort?**

- a) in-place algorithm
- b) pivot position can be changed**
- c) adaptive sorting algorithm
- d) can be implemented as a stable sort