Data Structure Operations Cheat Sheet

Data Structure Name	Average Case Time Complexity				Worst Case Time Complexity				Space Complexity
	Accessing n^{th} element	Search	Insertion	Deletion	Accessing n^{th} element	Search	Insertion	Deletion	Worst Case
Arrays	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stacks	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queues	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Binary Trees	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)
Binary Search Trees	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)	O(n)	O(n)	O(n)	O(n)	O(n)
Balanced Binary Search Trees	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)	O(*logn*)
Hash Tables	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)

Note: For best case operations, the time complexities are O(1).

Sorting Algorithms Cheat Sheet

Sorting Algorithm Name	Time Complexity			Space Complexity	Is Stable?	Sorting Class Type	Remarks
	Best Case	Average Case	Worst Case	Worst Case			
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)	Yes	Comparison	Not a preferred sorting algorithm.
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)	Yes	Comparison	In the best case (already sorted), every insert requires constant time
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	Yes	Comparison	Even a perfectly sorted array requires scanning the entire array
Merge Sort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)	O(n)	Yes	Comparison	On arrays, it requires O(n) space; and on linked lists, it requires constant space
Heap Sort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)	O(1)	No	Comparison	By using input array as storage for the heap, it is possible to achieve constant space
Quick Sort	O(*nlogn*)	O(*nlogn*)	$O(n^2)$	O(*logn*)	No	Comparison	Randomly picking a pivot value can help avoid worst case scenarios such as a perfectly sorted array.
Tree Sort	O(*nlogn*)	O(*nlogn*)	$O(n^2)$	O(n)	Yes	Comparison	Performing inorder traversal on the balanced binary search tree.
Counting Sort	$O(n + k)$	$O(n + k)$	$O(n + k)$	O(*k*)	Yes	Linear	Where *k* is the range of the non-negative key values.
Bucket Sort	$O(n + k)$	$O(n + k)$	$O(n^2)$	O(n)	Yes	Linear	Bucket sort is stable, if the underlying sorting algorithm is stable.
Radix Sort	O(*dn*)	O(*dn*)	O(*dn*)	$O(d + n)$	Yes	Linear	Radix sort is stable, if the underlying sorting algorithm is stable.

Data Structures

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Indexing	Search	Insertion	Deletion	Indexing	Search	Insertion	Deletion	
Basic Array	O(1)	O(n)	-	-	O(1)	O(n)	-	-	O(n)
Dynamic Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	-	O(1)	O(1)	O(1)	-	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	-	O(log(n))	O(log(n))	O(log(n))	-	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	-	O(log(n))	O(log(n))	O(log(n))	-	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

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 ∧D)/(E – F)+G

a) (A B D ∧ + E F – / G +)

b) (A B D +∧ E F – / G +)

c) (A B D ∧ + E F/- G +)

d) (A B D E F + ∧ / – 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 X c – d ^ e ^ f is?

a) abc X+ def ^^ –
b) abc X+ de^f^ –
c) ab+c Xd – e ^f^
d) -+aXbc^ ^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) θ(n)
d) θ(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) 0(n^2)
b) 0(n)
c) 0(1)
d) 0(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) Top = MAX + 1
b) Top = MAX - 1
c) 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(key)=key mod 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?

0.91
1.63
2.44
3.45
4.79
5.18
6.55

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

log28 + 1 = 3 + 1 = 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(n2)
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 in 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(logn)

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(logn)

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(logn)

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(nlogn)
b) O(logn)
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(nlogn)

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^2)

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(logn)

b) O(nlogn)

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(nlogn)

c) O(logn)

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(nlogn)

b) O(logn)

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(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
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) ^

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.

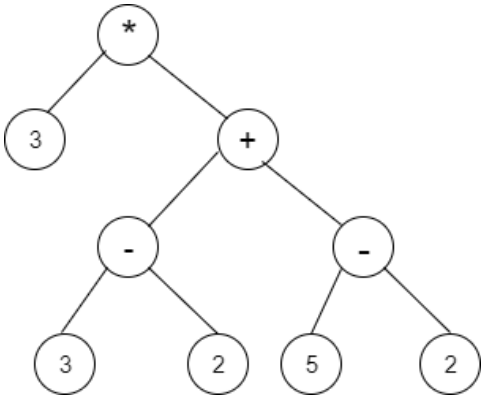
a) 6

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 '\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 -*+abcd = 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.

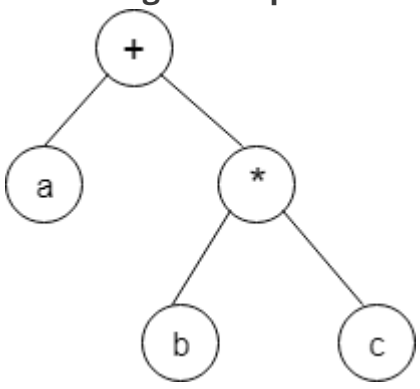
a) 61

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) ^^^ABCD
b) ^A^B^CD
c) ABCD^^^
d) AB^C^D

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

```
a+b-c/d&e|f
```

a) |&-+ab/cdef
b) &|-+ab/cdef
c) |&-ab+/cdef
d) |&-+/abcdef

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

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

a) -+a*/^bcdef
b) -+a*/bc^def
c) -+a*b/c^def
d) -a+*/bc^def

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|&bc
b) &|abc
c) |a&bc
d) ab&|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)
```

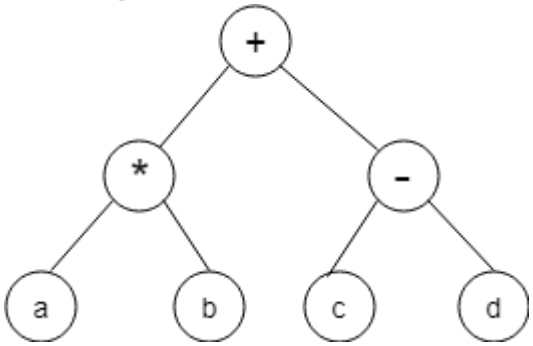
a) ab*cdef/^*g-h+
b) abcdef^/*g*h*+
c) abcd*^ed/g*-h*+
d) abc*de^fg/*-*h+

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(logn) 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) floor(w-1/2)

b) 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 seperate 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) find_size(root_node–>left_node) + 1 + find_size(root_node–>right_node)
b) find_size(root_node–>left_node) + find_size(root_node–>right_node)
c) find_size(root_node–>right_node) – 1
d) find_size(root_node–>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
```

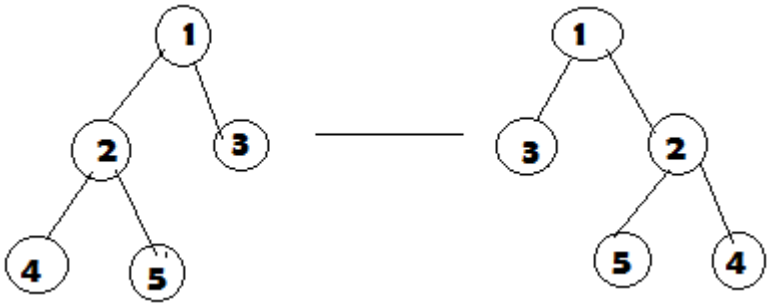
a) code for having recursive calls to either only left tree or right trees or to both subtrees depending on their existence

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

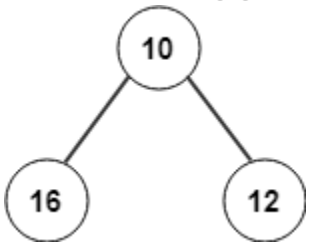
c) printing ancestors of a node passed as argument

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

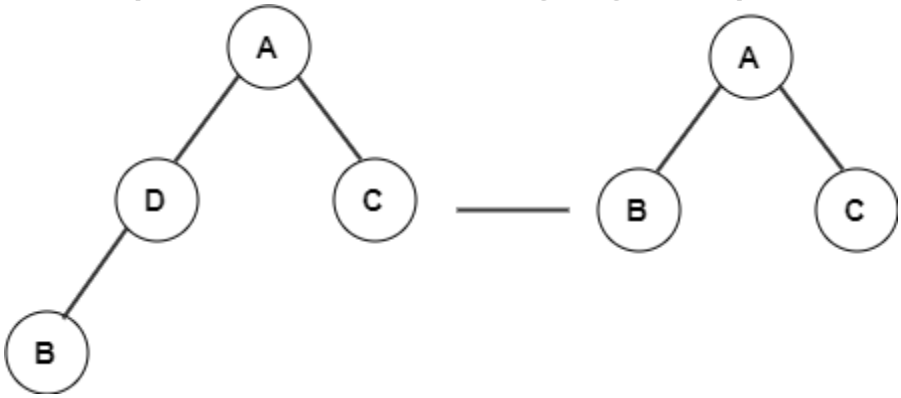
251. What is the traversal strategy used in the binary tree?

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(√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

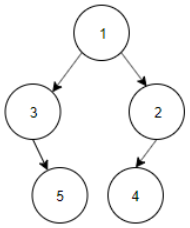
b) Right subtrees are visited before left subtrees

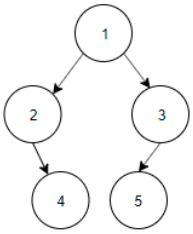
c) Root node is visited before left subtree

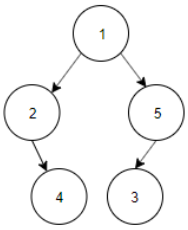
d) Root node is visited before right subtree

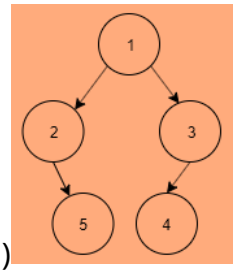
261. Construct a binary tree using the following data.

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.

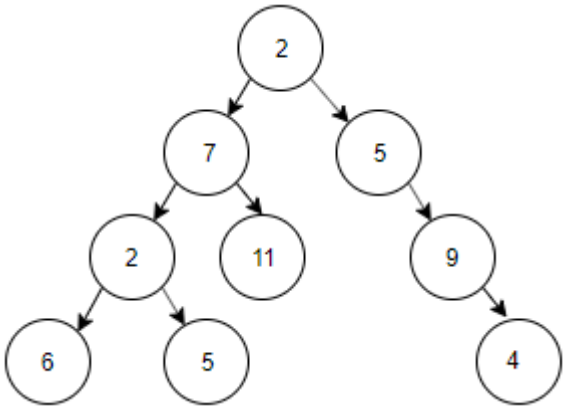
a) 

b) 

c) 

d) 

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(logn)

d) O(nlogn)

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(nlogd)
c) O(logd)
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

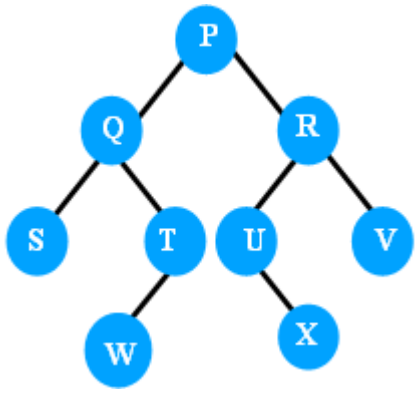
d) O P L M N Q T

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.



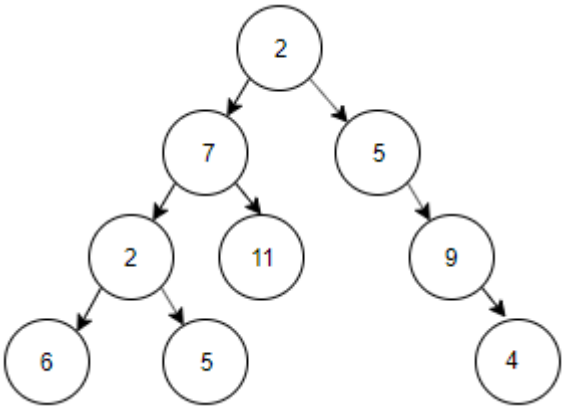
a) P Q R S T U V W X

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.



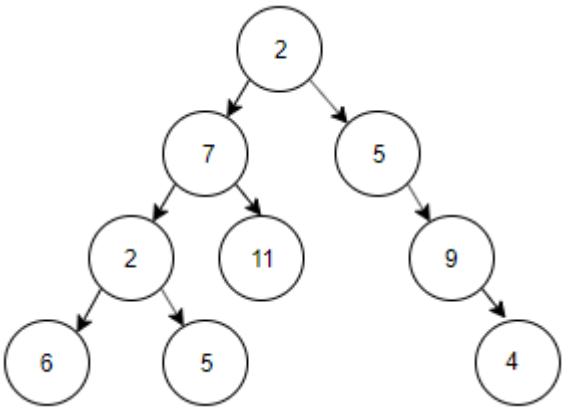
a) 6, 2, 5, 7, 11, 2, 5, 9, 4

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(nlogd)
c) O(logd)
d) O(d)

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

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

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(loglogn)

b) h = O(nlogn)

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 ≥ 0, there are no more than 2k nodes in level k

b) Let T be a binary tree with λ levels. Then T has no more than $2^{\lambda - 1}$ nodes

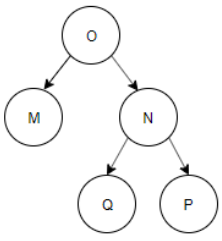
c) Let T be a binary tree with N nodes. Then the number of levels is at least ceil(log (N + 1))

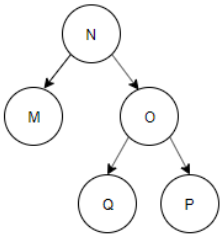
d) Let T be a binary tree with N nodes. Then the number of levels is at least floor(log (N + 1))

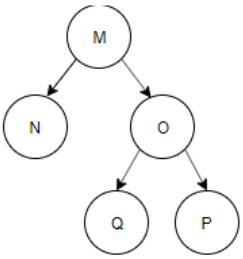
295. Construct a binary tree by using postorder and inorder sequences given below.

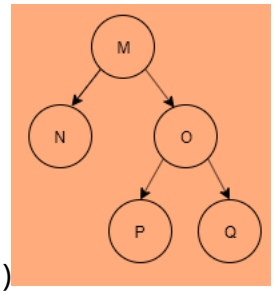
Inorder: N, M, P, O, Q

Postorder: N, P, Q, O, M

a) 

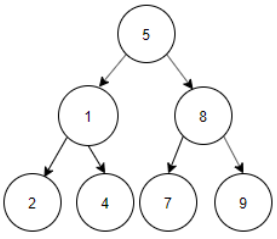
b) 

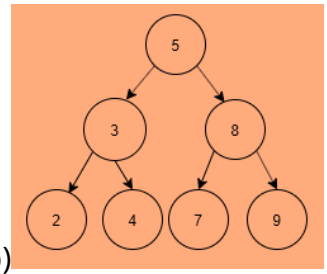
c) 

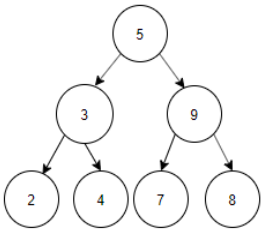
d) 

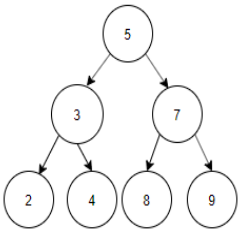
296. Construct a binary search tree by using postorder sequence given below.

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

a) 

b) 

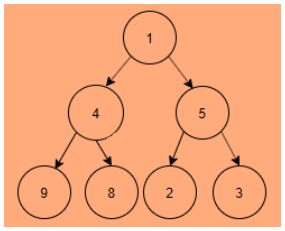
c) 

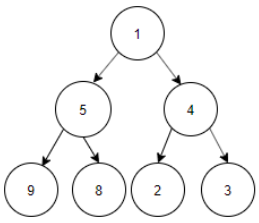
d) 

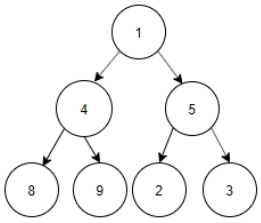
297. Construct a binary tree using inorder and level order traversal given below.

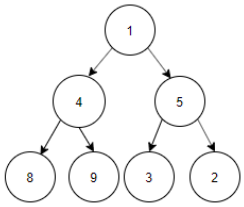
Inorder Traversal: 3, 4, 2, 1, 5, 8, 9

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

a) 

b) 

c) 

d) 

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(logn), O(logn)

c) O(logn), O(n)

d) O(n), O(logn)

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

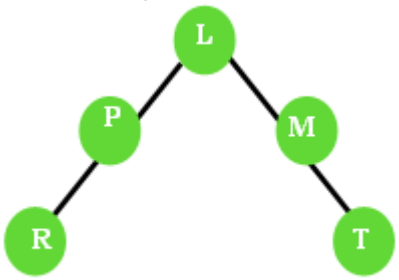
c) 6

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

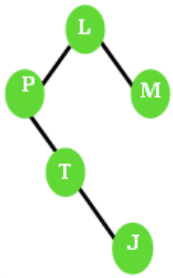
305. A binary tree is balanced if the difference between left and right subtree of every node is not more than ___

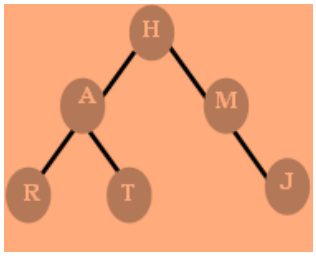
a) 1
b) 3
c) 2
d) 0

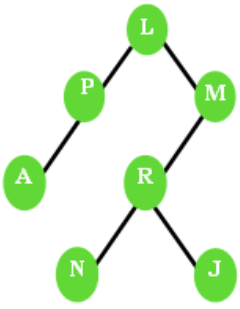
306. Which of the following tree data structures is not a balanced binary tree?

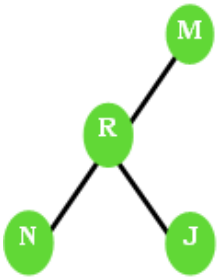
a) AVL tree
b) Red-black tree
c) Splay tree
d) B-tree

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

a) 

b) 

c) 

d) 

308. Balanced binary tree with n items allows the lookup of an item in ____ worst-case time.

a) O(log n)

b) O(nlog 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(mlog 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

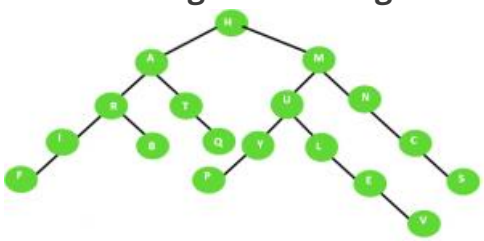
d) construction of the tree takes less time than binary heap

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(logn)
c) O(nlogn)
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(logn)

c) O(nlogn)

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(nlogn)
b) O(logn)
c) O(n)
d) O(1)

337. What is the worst case for linear search?

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

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

a) O(nlogn), O(logn)
b) O(logn), O(nlogn)
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 serach(recursive) vs linear search(Iterative)?

a) Both execute at same speed

b) Linear search(recursive) is faster

c) Linear search(Iterative) is faster

d) Cant 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(logn)

c) $O(n^2)$

d) O(nx)

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(logn)

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(nlogn)

b) O(logn)

c) O(n)

d) $O(n^2)$

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

a) O(nlogn)

b) O(logn)

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(nlogn)
b) O(logn)
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(nlogn)

b) $O(n^2)$

c) O(n)

d) O(logn)

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) || (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^{th} 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(logn) 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(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)

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

a) O(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
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(nlogn)

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)<=?

a) T(n) <= 2 T(n/4) + cn

b) T(n) <= T(n/4) + T(3n/4) + cn

c) T(n) <= 2 T(3n/4) + cn

d) T(n) <= 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(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
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(nlogn)
b) O(logn)
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

Q 1 - What is the worst case time complexity of linear search algorithm?

A - O(1)

B - O(n)

C - O(log n)

D - $O(n^2)$

Q 2 - Which of the following asymptotic notation is the worst among all?

A - O(n+9378)

B - $O(n^3)$

C - $n^{O(1)}$

D - $2^{O(n)}$

Q 3 - If the array is already sorted, which of these algorithms will exhibit the best performance

A - Merge Sort

B - Insertion Sort

C - Quick Sort

D - Heap Sort

Q 4 - An algorithm is

A - a piece of code to be executed.

B - a loosely written code to make final code.

C - a step by step procedure to solve problem.

D - all of the above.

Q 5 - In binary heap, whenever the root is removed then the rightmost element of last level is replaced by the root. Why?

A - It is the easiest possible way.

B - To make sure that it is still complete binary tree.

C - Because left and right subtree might be missing.

D - None of the above!

Q 6 - All possible spanning trees of graph G

A - have same number of edges and vertices.

B - have same number of edges and but not vertices.

C - have same number of vertices but not edges.

D - depends upon algorithm being used.

Q 7 - From a complete graph, by removing maximum _____ edges, we can construct a spanning tree.

A - e-n+1

B - n-e+1

C - n+e-1

D - e-n-1

Q 8 - Which of the following algorithm does not divide the list −

A - linear search

B - binary search

C - merge sort

D - quick sort

Q 9 - What is the worst case run-time complexity of binary search algorithm?

A - $O(n^2)$

B - $O(n^{\log n})$

C - $O(n^3)$

D - O(n)

Q 10 - A complete graph can have

A - n^2 spanning trees

B - n^{n-2} spanning trees

C - n^{n+1} spanning trees

D - n^n spanning trees

Q 11 - Which one of the below is not divide and conquer approach?

A - Insertion Sort

B - Merge Sort

C - Shell Sort

D - Heap Sort

Q 12 - Prefix notation is alsow known as

A - Reverse Polish Notation

B - Reverse Notation

C - Polish Reverse Notation

D - Polish Notation

Q 13 - In order traversal of binary search tree will produce −

A - unsorted list

B - reverse of input

C - sorted list

D - none of the above

Q 14 - In a min-heap:

A - parent nodes have values greater than or equal to their child

B - parent nodes have values less than or equal to their child

C - both statements are true

D - both statements are wrong

Q 15 - A procedure that calls itself is called

A - illegal call

B - reverse polish

C - recursive

D - none of the above

Q 16 - For a binary search algorithm to work, it is necessary that the array (list) must be

A - sorted

B - unsorted

C - in a heap

D - popped out of stack

Q 17 - push() and pop() functions are found in

A - queues

B - lists

C - stacks

D - trees

Q 18 - Queue data structure works on

A - LIFO

B - FIFO

C - FILO

D - none of the above

Q 19 - Maximum number of nodes in a binary tree with height k, where root is height 0, is

A - $2^k - 1$

B - $2^{k+1} - 1$

C - $2^{k-1} + 1$

D - $2^k - 1$

Q 20 - Which one of the below mentioned is linear data structure −

A - Queue

B - Stack

C - Arrays

D - All of the above

Q 21 - What data structure is used for depth first traversal of a graph?

A - queue

B - stack

C - list

D - none of the above

Q 22 - What data structure is used for breadth first traversal of a graph?

A - queue

B - stack

C - list

D - none of the above

Q 23 - What data structure can be used to check if a syntax has balanced paranthesis ?

A - queue

B - tree

C - list

D - stack

Q 24 - Postfix expression is just a reverse of prefix expression.

A - True

B - False

Q 25 - Stack is used for

A - CPU Resource Allocation

B - Breadth First Traversal

C - Recursion

D - None of the above

Q 26 - A circular linked list can be used for

A - Stack

B - Queue

C - Both Stack & Queue

D - Neither Stack or Queue

Q 27 - A linked-list is a dynamic structure

A - true

B - false

Q 28 - Minimum number of moves required to solve a *Tower of Hanoi* puzzle is

A - 2^{n2}

B - 2^{n-1}

C - $2^n - 1$

D - 2n - 1

Q 29 - Which of the following is an example of dynamic programming approach?

A - Fibonacci Series

B - Tower of Hanoi

C - Dijkstra Shortest Path

D - All of the above

Q 30 - The following formula will produce

$$F_n = F_{n-1} + F_{n-2}$$

A - Armstrong Number

B - Fibonacci Series

C - Euler Number

D - Prime Number

Q 31 - Minimum number of queues required for priority queue implementation?

A - 5

B - 4

C - 3

D - 2

Q 32 - Quick sort algorithm is an example of

A - Greedy approach

B - Improved binary search

C - Dynamic Programming

D - Divide and conquer

Q 33 - Which of the following asymptotic notation is the worst among all?

A - O(n+9378)

B - $O(n^3)$

C - $n^{O(1)}$

D - $2^{O(n)}$

Q 34 - The following formular is of

```
left_subtree (keys)  ≤  node (key)  ≤  right_subtree (keys)
```

A - Binary Tree

B - Complete Binary Tree

C - Binary Search Tree

D - All of the above

Q 35 - Travelling salesman problem is an example of

A - Dynamic Algorithm

B - Greedy Algorithm

C - Recursive Approach

D - Divide & Conquer

Q 36 - Find the odd out

A - Prim's Minimal Spanning Tree Algorithm

B - Kruskal's Minimal Spanning Tree Algorithm

C - Floyd-Warshall's All pair shortest path Algorithm

D - Dijkstra's Minimal Spanning Tree Algorithm

Q 37 - Which of the following searching techniques do not require the data to be in sorted form

A - Binary Search

B - Interpolation Search

C - Linear Search

D - All of the above

Q 38 - Minimum number of spanning tree in a connected graph is

A - n

B - n^{n-1}

C - 1

D - 0

Q 39 - Visiting root node after visiting left and right sub-trees is called

A - In-order Traversal

B - Pre-order Traversal

C - Post-order Traversal

Q 40 - Binary search tree has best case run-time complexity of O(log n). What could the worst case?

A - O(n)

B - $O(n^2)$

C - $O(n^3)$

D - None of the above

Q 41 - The minimum number of edges required to create a cyclid graph of n vertices is

A - n

B - n - 1

C - n + 1

D - 2n

Q 42 - Maximum degree of any vertex in a simple graph of vertices n is

A - 2n - 1

B - n

C - n + 1

D - n - 1

Q 43 - What could be the worst case height of an AVL tree?

A - 0.97 log n

B - 2.13 log n

C - 1.44 log n

D - n^2 log n

Q 44 - What is not true about insertion sort?

A - Exhibits the worst case performance when the initial array is sorted in reverse order.

B - Worst case and average case performance is $O(n^2)$

C - Can be compared to the way a card player arranges his card from a card deck.

D - None of the above!

Q 45 - Which of the following algorithm is not stable?

A - Bubble Sort

B - Quick Sort

C - Merge Sort

D - Insertion Sort

Q 46 - If the array is already sorted, which of these algorithms will exhibit the best performance

A - Merge Sort

B - Insertion Sort

C - Quick Sort

D - Heap Sort

Q 47 - Which of the following is example of in-place algorithm?

A - Bubble Sort

B - Merge Sort

C - Insertion Sort

D - All of the above

Q 48 - Graph traversal is different from a tree traversal, because

A - trees are not connected.

B - graphs may have loops.

C - trees have root.

D - None is true as tree is a subset of graph.

Q 49 - Which method can find if two vertices x & y have path between them?

A - Depth First Search

B - Breadth First Search

C - Both A & B

D - None A or B

Q 50 - Time complexity of Depth First Traversal of is

A - Θ(|V|+|E|)

B - Θ(|V|)

C - Θ(|E|)

D - Θ(|V|*|E|)

Q 51 - An algorithm is

A - a piece of code to be executed.

B - a loosely written code to make final code.

C - a step by step procedure to solve problem.

D - all of the above.

Q 52 – A priory algorithm analysis does not include −

A - Time Complexity

B - Space Complexity

C - Program Complexity

D - None of the above!

Q 53 - Which of the below given series is Non-Increasing Order −

A - 1, 3, 4, 6, 8, 9

B - 9, 8, 6, 4, 3, 1

C - 9, 8, 6, 3, 3, 1

D - 1, 3, 3, 6, 8, 9

Q 54 - Which of the following has search efficiency of O(1) −

A - Tree

B - Heap

C - Hash Table

D - Linked-List

Q 55 - After each iteration in bubble sort

A - at least one element is at its sorted position.

B - one less comparison is made in the next iteration.

C - Both A & B are true.

D - Neither A or B are true.

Q 56 - What about recursion is true in comparison with iteration?

A - very expensive in terms of memory.

B - low performance.

C - every recursive program can be written with iteration too.

D - all of the above are true!

1) How can we describe an array in the best possible way?

a. The Array shows a hierarchical structure.
b. Arrays are immutable.
c. Container that stores the elements of similar types
d. The Array is not a data structure

2) Which of the following is the correct way of declaring an array?

a. int javatpoint[10];
b. int javatpoint;
c. javatpoint{20};
d. array javatpoint[10];

4) Which of the following is the advantage of the array data structure?

a. Elements of mixed data types can be stored.
b. Easier to access the elements in an array
c. Index of the first element starts from 1.
d. Elements of an array cannot be sorted

5) Which of the following highly uses the concept of an array?

a. Binary Search tree
b. Caching
c. Spatial locality
d. Scheduling of Processes

6) Which of the following is the disadvantage of the array?

a. Stack and Queue data structures can be implemented through an array.
b. Index of the first element in an array can be negative
c. Wastage of memory if the elements inserted in an array are lesser than the allocated size
d. Elements can be accessed sequentially.

7) What is the output of the below code?

```
#include <stdio.h>
    int main()
    {
      int arr[5]={10,20,30,40,50};
      printf("%d", arr[5]);

       return 0;
    }
```

a. Garbage value
b. 10
c. 50
d. None of the above

8) Which one of the following is the size of int arr[9] assuming that int is of 4 bytes?

a. 9
b. 36
c. 35
d. None of the above

9) Which one of the following is the process of inserting an element in the stack?

a. Insert
b. Add
c. Push
d. None of the above

10) When the user tries to delete the element from the empty stack then the condition is said to be a ____

a. Underflow
b. Garbage collection
c. Overflow
d. None of the above

11) If the size of the stack is 10 and we try to add the 11th element in the stack then the condition is known as___

a. Underflow
b. Garbage collection
c. Overflow
d. None of the above

12) Which one of the following is not the application of the stack data structure

a. String reversal
b. Recursion
c. Backtracking
d. Asynchronous data transfer

13) Which data structure is mainly used for implementing the recursive algorithm?

a. Queue
b. Stack
c. Binary tree
d. Linked list

14) Which data structure is required to convert the infix to prefix notation?

a. Stack
b. Linked list
c. Binary tree
d. Queue

15) Which of the following is the infix expression?

a. A+B*C
b. +A*BC
c. ABC+*
d. None of the above

16) Which of the following is the prefix form of A+B*C?

a. A+(BC*)
b. +AB*C
c. ABC+*
d. +A*BC

17) Which of the following is not the correct statement for a stack data structure?

a. Arrays can be used to implement the stack
b. Stack follows FIFO
c. Elements are stored in a sequential manner
d. Top of the stack contains the last inserted element

18) If the elements '1', '2', '3' and '4' are added in a stack, so what would be the order for the removal?

a. 1234
b. 2134
c. 4321
d. None of the above

19) What is the outcome of the prefix expression +, -, *, 3, 2, /, 8, 4, 1?

a. 12
b. 11
c. 5
d. 4

20) The minimum number of stacks required to implement a stack is __

a. 1
b. 3
c. 2
d. 5

Answer: c

21) Which one of the following node is considered the top of the stack if the stack is implemented using the linked list?

a. First node
b. Second node
c. Last node
d. None of the above

22) Consider the following stack implemented using stack.

```
1. #define SIZE 11
2. struct STACK
3. {
4.    int arr[SIZE];
5.    int top=-1;
6. }
```

What would be the maximum value of the top that does not cause the overflow of the stack?

a. 8
b. 9
c. 11
d. 10

23) What is another name for the circular queue among the following options?

a. Square buffer
b. Rectangle buffer
c. Ring buffer
d. None of the above

24) If the elements '1', '2', '3' and '4' are inserted in a queue, what would be order for the removal?

a. 1234
b. 4321
c. 3241
d. None of the above

25) A list of elements in which enqueue operation takes place from one end, and dequeue operation takes place from one end is__

a. Binary tree
b. Stack
c. Queue
d. Linked list

26) Which of the following principle does Queue use?

a. LIFO principle
b. FIFO principle
c. Linear tree
d. Ordered array

27) Which one of the following is not the type of the Queue?

a. Linear Queue
b. Circular Queue
c. Double ended Queue
d. Single ended Queue

28) Which one of the following is the overflow condition if linear queue is implemented using an array with a size MAX_SIZE?

a. rear = front
b. rear = front+1
c. rear=MAX_SIZE -1
d. rear = MAX_SIZE

29) Which one of the following is the overflow condition if a circular queue is implemented using array having size MAX?

a. rear= MAX-1
b. rear=MAX
c. front=(rear+1) mod max
d. None of the above

30) The time complexity of enqueue operation in Queue is __

a. O(1)
b. O(n)
c. O(logn)
d. O(nlogn)

31) Which of the following that determines the need for the Circular Queue?

a. Avoid wastage of memory
b. Access the Queue using priority
c. Follows the FIFO principle
d. None of the above

32) Which one of the following is the correct way to increment the rear end in a circular queue?

a. rear =rear+1
b. (rear+1) % max
c. (rear % max) + 1
d. None of the above

33) Consider the following code.

```
1.  int fun()
2.  {
3.     if(isEmpty())
4.     {
5.        return -10;
6.     }
7.    else
8.    {
9.       int n;
10.      n= q[front];
11.      front++;
12.      return n;
13.   }
14.
15. }
```

Which operation does the above code perform?

a. Enqueue
b. Dequeue
c. Return the front element
d. Both b and c

34) In the linked list implementation of queue, where will the new element be inserted?

a. At the middle position of the linked list
b. At the head position of the linked list
c. At the tail position of the linked list
d. None of the above

35) How many Queues are required to implement a Stack?

a. 3
b. 2
c. 1
d. 4

36) Which one of the following is not the application of the Queue data structure?

a. Resource shared between various systems
b. Data is transferred asynchronously
c. Load balancing
d. Balancing of symbols

37) Which of the following option is true if implementation of Queue is from the linked list?

a. In enqueue operation, new nodes are inserted from the beginning and in dequeue operation, nodes are removed from the end.
b. In enqueue operation, new nodes are inserted from the end and in dequeue operation, nodes are deleted from the beginning.
c. In enqueue operation, new nodes are inserted from the end and in dequeue operation, nodes are deleted from the end.
d. Both a and b

38) The necessary condition to be checked before deletion from the Queue is__

a. Overflow
b. Underflow
c. Rear value
d. Front value

39) Which data structure is the best for implementing a priority queue?

a. Stack
b. Linked list
c. Array
d. Heap

40) Which of the following principle is used if two elements in the priority queue have the same priority?

a. LIFO
b. FIFO
c. Linear tree
d. None of the above

41) Which of the following statement is not true regarding the priority queue?

a. Processes with different priority can be easily handled
b. Easy to implement
c. Deletion is easier
d. None of the above

42) A linear data structure in which insertion and deletion operations can be performed from both the ends is___

a. Queue
b. Deque
c. Priority queue
d. Circular queue

43) In the Deque implementation using singly linked list, what would be the time complexity of deleting an element from the rear end?

a. O(1)
b. $O(n^2)$
c. O(n)
d. O(nlogn)

44) Which of the following data structure allows you to insert the elements from both the ends while deletion from only one end?

a. Input-restricted queue
b. Output-restricted queue
c. Priority queue
d. None of the above

45) What would be the output after performing the following operations in a Deque?

1. Insertfront(10);
2. Insertfront(20);
3. Insertrear(30);
4. Insertrear(40);
5. Deletefront();
6. Insertfront(50);
7. Deleterear();
8. Display();

a. 10, 20, 30

b. 50, 10, 30

c. 40, 20, 30

d. None of the above

46) In a circular queue implementation using array of size 5, the array index starts with 0 where front and rear values are 3 and 4 respectively. Determine the array index at which the insertion of the next element will take place.

a. 5

b. 0

c. 1

d. 2

47) If circular queue is implemented using array having size MAX_SIZE in which array index starts with 0, front points to the first element in the queue, and rear points to the last element in the queue. Which one of the following conditions used to specify that the circular queue is empty?

a. Front=rear= -1

b. Front=rear=0

c. Front=rear+1

d. None of the above

48) Consider the implementation of the singly linked list having the head pointer only in the representation. Which of the following operations can be performed in O(1) time?

i) Deletion of the last node in the linked list
ii) Insertion at the front of the linked list
iii) Deletion of the first node in the linked list
iv) Insertion at the end of the linked list

a. ii
b. both ii and iii
c. both i and iv
d. both i and ii

49) What would be the time complexity if user tries to insert the element at the end of the linked list (head pointer is known)?

a. O(1)
b. O(n)
c. O(logn)
d. O(nlogn)

50) Which of the following is the time complexity to search an element in the linked list?

a. O(1)
b. O(n)
c. O(logn)
d. O(nlogn)

51) Consider the following code

```
1. struct node
2. {
3.    int data;
4.    struct node *next;
5. }
6. node ptr;
```

Which one of the following is the correct option to create a new node?

a. ptr= (node*)malloc(sizeof(node*))

b. ptr=(node)malloc(sizeof(node))

c. ptr=(node*)malloc(sizeof(node))

d. None of the above

52) Which of the following statement is not true about the doubly linked list?

a. We can traverse in both the directions.

b. It requires extra space

c. Implementation of doubly linked list is easier than the singly linked list

d. It stores the addresses of the next and the previous node

53) What is the maximum number of children that a node can have in a binary tree?

a. 3

b. 1

c. 4

d. 2

54) Which one of the following techniques is not used in the Binary tree?

a. Randomized traversal

b. Preorder traversal

c. Postorder traversal

d. Inorder traversal

55) Which of the following options is not true about the Binary Search tree?

a. The value of the left child should be less than the root node

b. The value of the right child should be greater than the root node.

c. The left and right sub trees should also be a binary search tree

d. None of the above

56) How can we define a AVL tree?

a. **A tree which is binary search tree and height balanced tree.**
b. A tree which is a binary search tree but unbalanced tree.
c. A tree with utmost two children
d. A tree with utmost three children

57) Why do we prefer Red Black tree over AVL tree?

a. Red Black trees are not strictly balanced
b. Red black tree requires lesser rotations than AVL tree.
c. AVL tree needs more space to store the balance factor.
d. **Both b and c**

58) Which of the following satisfies the property of the Red Black tree?

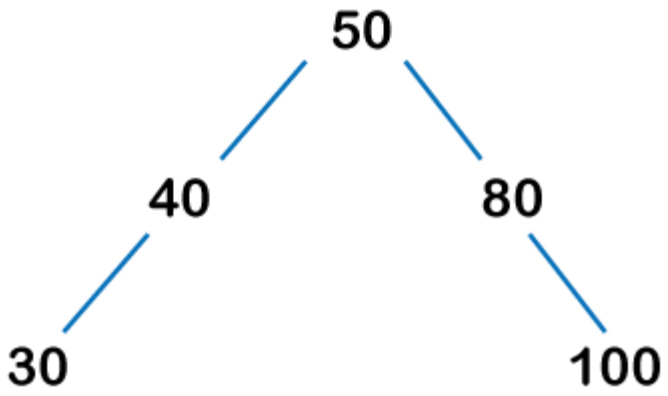
a. A tree which is a binary search tree but not strictly balanced tree.
b. A node must be either Red or Black in color and root node must be black.
c. A tree with maximum three children
d. **Both a and b**

59) What would be the color of newly created node while inserting a new element in a Red black tree?

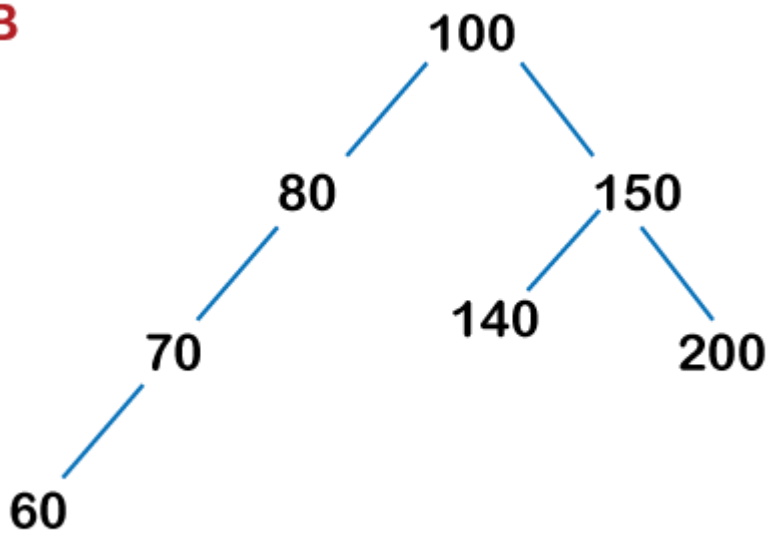
a. Black, if the new node is not a root node
b. Red, if the new node is not a root node
c. Black, if the new node is a root node
d. **Both b and c**

60) Identify the AVL tree among the following options?

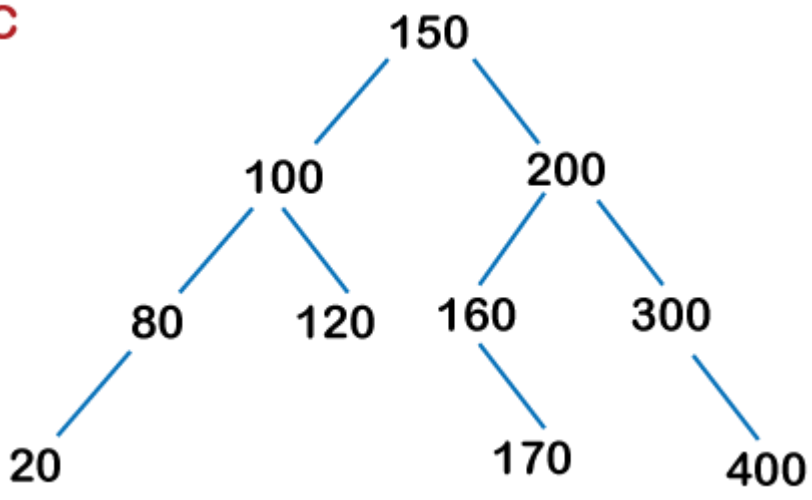
A



B



C



a. A

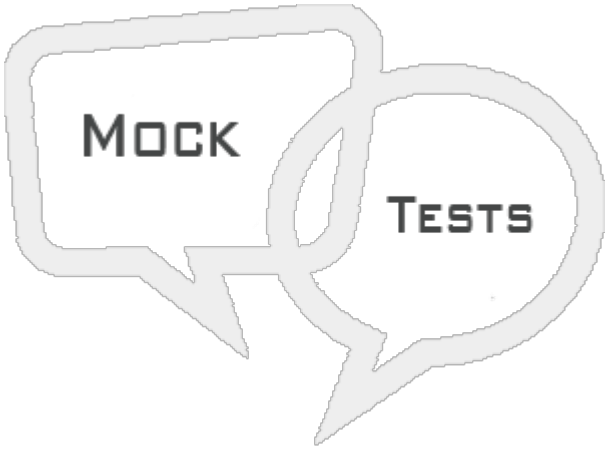
b. C

c. Both A and C

d. B

DATA STRUCTURES ALGORITHMS MOCK TEST

This section presents you various set of Mock Tests related to **Data Structures Algorithms**. You can download these sample mock tests at your local machine and solve offline at your convenience. Every mock test is supplied with a mock test key to let you verify the final score and grade yourself.



DATA STRUCTURES ALGORITHMS MOCK TEST III

Q 1 - What will be the running-time of Dijkstra's single source shortest path algorithm, if the graph GV, E is stored in form of adjacency list and binary heap is used −

A - O(|V|^2)

B - O$|V|log|V|$

C - O$|E| + |V|log|V|$

D - None of these

Q 2 - How many swaps are required to sort the given array using bubble sort - { 2, 5, 1, 3, 4}

A - 4

B - 5

C - 6

D - 7

Q 3 - Match the following −

(**1**) Bubble Sort	(**A**) On
(**2**) Shell Sort	(**B**) O(n^2)
(**3**) Selection Sort	(**C**) O$nlogn$

A - 1 → A, 2 → B, 3 → C

B - 1 → B, 2 → C, 3 → A

C - 1 → A, 2 → C, 3 → B

D - 1 → B, 2 → A, 3 → C

Q 4 - In context with time-complexity, find the odd out −

A - Deletion from Linked List.

B - Searching in Hash Table

C - Adding edge in Adjacency Matrix

D - Heapify a Binary Heap

Q 5 - In binary heap, whenever the root is removed then the rightmost element of last level is replaced by the root. Why?

A - It is the easiest possible way.

B - To make sure that it is still complete binary tree.

C - Because left and right subtree might be missing.

D - None of the above!

Q 6 - Time required to merge two sorted lists of size m and n, is

A - O$m|n$

B - O$m+n$

C - O$mlogn$

D - O$nlogm$

Q 7 - The number of binary trees with 3 nodes which when traversed in post order gives the sequence A,B,C is ?

A - 3

B - 4

C - 5

D - 6

Q 8 - Quick sort running time depends on the selection of

A - size of array

B - pivot element

C - sequence of values

D - none of the above!

Q 9 - Which of the below given sorting techniques has highest best-case runtime complexity −

A - quick sort

B - selection sort

C - insertion sort

D - bubble sort

Q 10 - Which of the below mentioned sorting algorithms are not stable?

A - Selection Sort

B - Bubble Sort

C - Merge Sort

D - Insertion Sort

Q 11 - If queue is implemented using arrays, what would be the worst run time complexity of queue and dequeue operations?

A - $O(n)$, $O(n)$

B - $O(n)$, $O(1)$

C - $O(1)$, $O(n)$

D - $O(1)$, $O(1)$

Q 12 - A queue data-structure can be used for −

A - expression parsing

B - recursion

C - resource allocation

D - all of the above

Q 13 - The Θ notation in asymptotic evaluation represents −

B - Base case

C - Average case

D - Worst case

A - NULL case

Q 14 - Which of these alogrithmic approach tries to achieve localized optimum solution −

A - Greedy approach

B - Divide and conquer approach

C - Dynamic approach

D - All of the above

Q 15 - Which of the following uses memoization?

A - Greedy approach

B - Divide and conquer approach

C - Dynamic programming approach

D - None of the above!

Q 16 - Index of arrays in C programming langauge starts from

A - 0

B - 1

C - either 0 or 1

D - undefined

Q 17 - In doubly linked lists

A - a pointer is maintained to store both next and previous nodes.

B - two pointers are maintained to store next and previous nodes.

C - a pointer to self is maintained for each node.

D - none of the above.

Q 18 - `node.next -> node.next.next;` **will make**

A - node.next inaccessible

B - node.next.next inaccessible

C - this node inaccessible

D - none of the above

Q 19 - Linked list search complexity is

A - $O1$

B - On

C - $Ologn$

D - $Ologlogn$

Q 20 - Which of the following is not possible with an array in C programming langauge –

A - Declaration

B - Definition

C - Dynamic Allocation

D - Array of strings

Q 21 - In C programming, when we remove an item from bottom of the stack, then −

A - The stack will fall down.

B - Stack will rearranged items.

C - It will convert to LIFO

D - This operation is not allowed.

Q 22 - Program with highest run-time complexity is

A - Tower of Hanoi

B - Fibonacci Series

C - Prime Number Series

D - None of the above

Q 23 - Tower of hanoi is a classic example of

A - divide and conquer

B - recursive approach

C - B but not A

D - Both A & B

Q 24 - Which of the following algorithm cannot be desiged without recursion −

A - Tower of Hanoi

B - Fibonacci Series

C - Tree Traversal

D - None of the above

Q 25 - If there's no base criteria in a recursive program, the program will

A - not be executed.

B - execute until all conditions match.

C - execute infinitely.

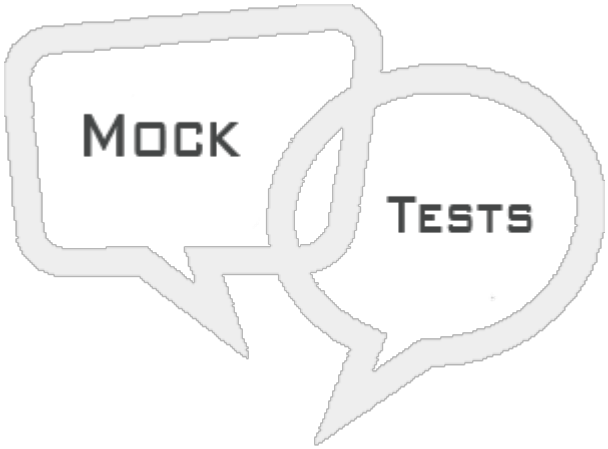
D - obtain progressive approach.

ANSWER SHEET

Question Number	Answer Key
1	C
2	A
3	B
4	D
5	B
6	B
7	C
8	B
9	B
10	A
11	D
12	C
13	B
14	A
15	C
16	A
17	B
18	A
19	B
20	C
21	D
22	A
23	D
24	D
25	C

DATA STRUCTURES ALGORITHMS MOCK TEST

This section presents you various set of Mock Tests related to **Data Structures Algorithms**. You can download these sample mock tests at your local machine and solve offline at your convenience. Every mock test is supplied with a mock test key to let you verify the final score and grade yourself.



DATA STRUCTURES ALGORITHMS MOCK TEST IV

Q 1 - Recursion uses more memory space than iteration because

A - it uses stack instead of queue.

B - every recursive call has to be stored.

C - both A & B are true.

D - None of the above are true.

Q 2 - Heap is an example of

A - complete binary tree

B - spanning tree

C - sparse tree

D - binary search tree

Q 3 - In a min heap

A - minimum values are stored.

B - child nodes have less value than parent nodes.

C - parent nodes have less value than child nodes.

D - maximum value is contained by the root node.

Q 4 - In the deletion operation of max heap, the root is replaced by

A - next available value in the left sub-tree.

B - next available value in the right sub-tree.

C - any random value from the heap.

D - last element of the last level

Q 5 - All possible spanning trees of graph G

A - have same number of edges and vertices.

B - have same number of edges and but not vertices.

C - have same number of vertices but not edges.

D - depends upon algorithm being used.

Q 6 - From a complete graph, by removing maximum _____ edges, we can construct a spanning tree.

A - **e-n+1**

B - **n-e+1**

C - **n+e-1**

D - **e-n-1**

Q 7 - If we choose Prim's Algorithm for uniquely weighted spanning tree instead of Kruskal's Algorithm, then

A - we'll get a different spanning tree.

B - we'll get the same spanning tree.

C - spanning will have less edges.

D - spanning will not cover all vertices.

Q 8 - Re-balancing of AVL tree costs

A - $O1$

B - $Ologn$

C - On

D - $O(n^2)$

Q 9 - A balance factor in AVL tree is used to check

A - what rotation to make.

B - if all child nodes are at same level.

C - when the last rotation occured.

D - if the tree is unbalanced.

Q 10 - Binary search tree is an example of complete binary tree with special attributes.

A - BST does not care about complete binary tree properties.

B - BST takes care of complete binary tree properties.

C - It depends upon the input.

D - None of the above.

Q 11 - The following sorting algorithms maintain two sub-lists, one sorted and one to be sorted −

A - Selection Sort

B - Insertion Sort

C - Merge Sort

D - both A &am; B

Q 12 - If locality is a concern, you can use _____ to traverse the graph.

A - Breadth First Search

B - Depth First Search

C - Either BFS or DFS

D - None of the above!

Q 13 - Access time of a binary search tree may go worse in terms of time complexity upto

A - $O(n^2)$

B - $Onlogn$

C - On

D - $O1$

Q 14 - Shell sort uses

A - insertion sort

B - merge sort

C - selection sort

D - quick sort

Q 15 - A pivot element to partition unsorted list is used in

A - Merge Sort

B - Quick Sort

C - Insertion Sort

D - Selection Sort

Q 16 - A stable sorting alrithm −

A - does not crash.

B - does not run out of memory.

C - does not change the sequence of appearance of elements.

D - does not exists.

Q 17 - An adaptive sorting algorithm −

A - adapts to new computers.

B - takes advantage of already sorted elements.

C - takes input which is already sorted.

D - none of the above.

Q 18 - Interpolation search is an improved variant of binary search. It is necessary for this search algorithm to work that −

A - data collection should be in sorted form and equally distributed.

B - data collection should be in sorted form and but not equally distributed.

C - data collection should be equally distributed but not sorted.

D - None of the above.

Q 19 - If the data collection is in sorted form and equally distributed then the run time complexity of interpolation search is −

A - On

B - O1

C - O$logn$

D - O$log(logn)$

Q 20 - Which of the following algorithm does not divide the list −

A - linear search

B - binary search

C - merge sort

D - quick sort

Q 21 - The worst case complexity of binary search matches with −

A - interpolation search

B - linear search

C - merge sort

D - none of the above

Q 22 - Apriori analysis of an algorithm assumes that −

A - the algorithm has been tested before in real environment.

B - all other factors like CPU speed are constant and have no effect on implementation.

C - the algorithm needs not to be practical.

D - none of the above.

Q 23 - Aposterior analysis are more accurate than apriori analysis because −

A - it contains the real data.

B - it assumes all other factors to be dynamic.

C - it assumes all other factors to be constant.

D - it is a result of reverse-engineering.

Q 24 - Project scheduling is an example of

A - greedy programming

B - dynamic programming

C - divide and conquer

D - none of the above.

Q 25 - In conversion from prefix to postfix using stack data-structure, if operators and operands are pushed and popped exactly once, then the run-time complexity is −

A - $O1$

B - On

C - $Ologn$

D - $O(n^2)$

ANSWER SHEET

Question Number	Answer Key
1	B
2	A

3	C
4	D
5	A
6	A
7	B
8	B
9	D
10	A
11	D
12	B
13	C
14	A
15	B
16	C
17	B
18	A
19	D
20	A
21	B
22	B
23	A
24	B
25	B

1. How is an array initialized in C language?

A. **int a[3] = {1, 2, 3};**
B. int a = {1, 2, 3};
C. int a[] = new int[3]
D. int a(3) = [1, 2, 3];

2.Which of the following is a linear data structure?

A. **Array**
B. AVL Trees
C. Binary Trees
D. Graphs

3.How is the 2nd element in an array accessed based on pointer notation?

A. *a + 2
B. ***(a + 2)**
C. *(*a + 2)
D. &(a + 2)

4. Which of the following is not the type of queue?

A. Priority queue
B. **Single-ended queue**
C. Circular queue
D. Ordinary queue

5. From following which is not the operation of data structure?

A. Operations that manipulate data in some way
B. Operations that perform a computation
C. **Operations that check for syntax errors**
D. Operations that monitor an object for the occurrence of a controlling event

6. What will be the output of the following code snippet?

```
void solve() {
   int a[] = {1, 2, 3, 4, 5};
   int sum = 0;
   for(int i = 0; i < 5; i++) {
       if(i % 2 == 0) {
           sum += a[i];
       }
   }
   cout << sum << endl;
}
```

A. 5
B. 15
C. **9**
D. 6

7. What will the output of the following code snippet?

```
void solve() {
   int a[] = {1, 2, 3, 4, 5};
   int sum = 0;
   for(int i = 0; i < 5; i++) {
       if(i % 2 == 0) {
           sum += *(a + i);
       }
       else {
           sum -= *(a + i);
       }
   }
   cout << sum << endl;
}
```

A. 2
B. 15
C. Syntax Error
D. 3

8. What is the disadvantage of array data structure?

A. The amount of memory to be allocated should be known beforehand.
B. Elements of an array can be accessed in constant time.
C. Elements are stored in contiguous memory blocks.
D. Multiple other data structures can be implemented using arrays.

9. How are String represented in memory in C?

A. An array of characters.
B. The object of some class.
C. Same as other primitive data types.
D. LinkedList of characters.

10. What is the output of the following code snippet?

```
void solve() {
   stack<int> s;
   s.push(1);
   s.push(2);
   s.push(3);
   for(int i = 1; i <= 3; i++) {
       cout << s.top() << " ";
       s.pop();
   }
}
```

A. 3 2 1
B. 1 2 3
C. 3
D. 1

11. Which of the following is the advantage of the array data structure?

A. Elements of mixed data types can be stored.
B. Easier to access the elements in an array
C. Index of the first element starts from 1.
D. Elements of an array cannot be sorted

12. What function is used to append a character at the back of a string in C++?

A. push_back()
B. append()
C. push()
D. insert()

13. Which one of the following is an application of queue data structure

A. When a resource is shared among multiple consumers.
B. When data is transferred asynchronously
C. Load Balancing
D. All of the above

14. When a pop() operation is called on an empty queue, what is the condition called?

A. Overflow
B. Underflow
C. Syntax Error
D. Garbage Value

15. What is the time complexity of the following code snippet in C++?

```
void solve() {
   string s = "scaler";
   int n = s.size();
   for(int i = 0; i < n; i++) {
       s = s + s[i];
   }
   cout << s << endl;
}
```

A. O(n)
B. O(n^2)
C. O(1)
D. O(log n)

16. Which of the following data structures can be used to implement queues?

A. Stack
B. Arrays
C. LinkedList
D. All of the Above

17. Which of the following data structures finds its use in recursion?

- **A. Stack**
- B. Arrays
- C. LinkedList
- D. Queues

18. Which of the following data structures allow insertion and deletion from both ends?

- A. Stack
- **B. Deque**
- C. Queue
- D. Strings

19. What will be the output of the following code snippet?

```
void solve() {
   deque<int> dq;
   for(int i = 1; i <= 5; i++) {
       if(i % 2 == 0) {
           dq.push_back(i);
       }
       else {
           dq.push_front(i);
       }
   }
   for(auto x: dq) {
       cout << x << " ";
   }
   cout << endl;
}
```

- A. 1 2 3 4 5
- B. 5 4 3 2 1
- C. 1 3 5 2 4
- **D. 5 3 1 2 4**

20. Which of the following sorting algorithms provide the best time complexity in the worst-case scenario?

- **A. Merge Sort**
- B. Quick Sort
- C. Bubble Sort
- D. Selection Sort

21. What is the maximum number of swaps that can be performed in the Selection Sort algorithm?

- **A. n - 1**
- B. n
- C. 1
- D. n - 2

22. Which of the following is a Divide and Conquer algorithm?

A. Bubble Sort
B. Selection Sort
C. Heap Sort
D. Merge Sort

23. What will be the best sorting algorithm, given that the array elements are small (<= 1e6)?

A. Bubble Sort
B. Merge Sort
C. Counting Sort
D. Heap Sort

24. Which of the following are applications of Topological Sort of a graph?

A. Sentence Ordering.
B. Course Scheduling.
C. OS Deadlock Detection.
D. All of the above.

25. Which of the following is known to be not an NP-Hard Problem?

A. Vertex Cover Problem.
B. 0/1 Knapsack Problem.
C. Maximal Independent Set Problem.
D. Travelling Salesman Problem.

26. Which of the following algorithms are used for string and pattern matching problems?

A. Z Algorithm
B. Rabin Karp Algorithm
C. KMP Algorithm
D. All of the above

27. Consider we have a function, getLCA(), which returns us the Lowest Common Ancestor between 2 nodes of a tree. Using this getLCA() function, how can we calculate the distance between 2 nodes, given that distance from the root, to each node is calculated?

A. dist(u) + dist(v) - 2 * dist(getLCA(u, v))
B. dist(u) + dist(v) + 2 * dist(getLCA(u, v))
C. dist(u) + dist(v)
D. dist(u) + dist(v) - dist(getLCA(u, v))

28. Which of the following algorithms are useful for processing queries on trees?

A. Centroid Decomposition.
B. Heavy Light Decomposition.
C. Both (A) and (B).
D. Neither (A) nor (B).

29. What will the output of the following code snippet be?

```
void solve() {
   vector<int> a = {1, 2, 3, 4, 5};
   sort(a.begin(), a.end(), [&](const int &x, const int &y) {
       return x % 2 < y % 2;
   });
   for(int x: a) {
       cout << x << " ";
   }
   cout << endl;
}
```

A. 1 2 3 4 5
B. 5 4 3 2 1
C. 1 3 5 2 4
D. 2 4 1 3 5

30. Consider the following code snippet:

```
void solve(vector<int> &a) {
   int queries;
   cin >> queries;
   while(queries--) {
       int type;
       cin >> type;
       if(type == 1) {
           int index, value;
           cin >> index >> value;
           update(a, index, value);
       }
       else {
           int l, r;
           cin >> l >> r;
           cout << getXOR(a, l, r) << endl;
       }
   }
}
```

The update() function updates the element at the given index in the array to some given value. The getXOR() function returns the XOR of the elements in the array a, in the range [l, r]. Which of the following data structures can perform the above tasks optimally?

A. Segment Trees.
B. Prefix XOR Arrays.
C. Tries.
D. Stacks.

31. What is the time complexity of the binary search algorithm?
A. O(n)
B. O(1)
C. O(logn)
D. O(n^2)

32. Kruskal's Algorithm for finding the Minimum Spanning Tree of a graph is a kind of a?
A. DP Problem.
B. Greedy Algorithm.
C. Adhoc Problem.
D. None of the above.

33. What will be the output of the following code snippet?

```
void solve() {
   string s = "00000001111111";
   int l = 0, r = s.size() - 1, ans = -1;
   while(l <= r) {
       int mid = (l + r) / 2;
       if(s[mid] == '1') {
           ans = mid;
           r = mid - 1;
       }
       else {
           l = mid + 1;
       }
   }
   cout << ans << endl;
}
```

A. 6
B. 7
C. 0
D. 1

34. Maps in C++ are implemented using which of the following data structures?
A. Red-Black Trees.
B. Binary Search Trees.
C. AVL Trees.
D. Hash Tables.

35. What will be the output of the following code snippet?

```
void solve() {
   int n = 24;
   int l = 0, r = 100, ans = n;
   while(l <= r) {
       int mid = (l + r) / 2;
       if(mid * mid <= n) {
           ans = mid;
           l = mid + 1;
       }
       else {
           r = mid - 1;
       }
   }
   cout << ans << endl;
}
```

A. 5
B. 4
C. 6
D. 3

36. What is the time complexity of the Sieve of Eratosthenes to check if a number is prime?

A. O(nlog(logn)) Precomputation, O(1) for check.
B. O(n) Precomputation, O(1) for the check.
C. O(n * logn) Precomputation, O(logn) for check.
D. O(n) Precomputation, O(logn) for check.

37. What will be the output of the following code snippet?

```
int search(int l, int r, int target, vector<int> &a) {
   int mid = (l + r) / 2;
   if(a[mid] == target) {
       return mid;
   }
   else if(a[mid] < target) {
       return search(mid + 1, r, target, a);
   }
   else {
       return search(0, mid - 1, target, a);
   }
}
void solve() {
   vector<int> a = {1, 2, 3, 4, 5};
   cout << search(0, 4, 4, a) << endl;
}
```

A. 3
B. 4
C. 0
D. 2

38. What is the best case time complexity of the binary search algorithm?

A. O(1)
B. O(n)
C. O(log2n)
D. O(n^2)

39. What is the time complexity to insert an element to the front of a LinkedList(head pointer given)?

A. O(n)
B. O(1)
C. O(logn)
D. O(n * logn)

40. What is the time complexity to insert an element to the rear of a LinkedList(head pointer given)?

A. O(n)
B. O(1)
C. O(logn)
D. O(n * logn)

41. What will be the value of "sum" after the following code snippet terminates?

```
void solve(ListNode* root) {
   /*
   The LinkedList is defined as:
   root-> val = value of the node
   root-> next = address of next element from the node
   The List is 1 -> 2 -> 3 -> 4 -> 5
   */
   int sum = 0;
   while(root -> next != NULL) {
       sum += root -> val;
       root = root -> next;
   }
   cout << sum << endl;
}
```

A. 15
B. 20
C. 5
D. 1

42. Which of the following can be done with LinkedList?

A. Implementation of Stacks and Queues
B. Implementation of Binary Trees
C. Implementation of Data Structures that can simulate Dynamic Arrays
D. All of the above

43. What is the information, which a LinkedList's Node must store?

A. The address of the next node if it exists
B. The value of the current node
C. Both (A) and (B)
D. None of the above

44. What is the maximum number of children a node can have in an n-ary tree?

A. 2
B. 0
C. 1
D. n

45. Worst case time complexity to access an element in a BST can be?

A. O(n)
B. O(n * logn)
C. O(1)
D. O(logn)

46. Which of the following represents the Postorder Traversal of a Binary Tree?

A. Left -> Right -> Root
B. Left -> Root -> Right
C. Right -> Left -> Root
D. Right -> Root -> Left

47. In what time complexity can we find the diameter of a binary tree optimally?

A. O(V + E)
B. O(V)
C. O(E)
D. O(V * logE)

48. Which of the following statements is true about AVL Trees?

A. The difference between the heights of left and right nodes cannot be more than 1.
B. The height of an AVL Tree always remains of the order of O(logn)
C. AVL Trees are a type of self-balancing Binary Search Trees.
D. All of the above.

49. What does the following code snippet calculate (edges represent the adjacency list representation of a graph)?

```
void solve(vector<vector<int>> edges) {
   int count = 0;
   for(auto x: edges) {
       for(auto y: x) {
           count += 1;
       }
   }
   cout << count / 2 << endl;
}
```

A. Calculates the number of edges in an undirected graph.
B. Calculates the number of nodes in a given graph.
C. Calculates the sum of degrees of all nodes in a given graph.
D. None of the above.

50. In a graph of n nodes and n edges, how many cycles will be present?

A. Exactly 1
B. At most 1
C. At most 2
D. Depends on the graph

51. A node in a tree, such that removing it splits the tree into forests, with size of each connected component being not greater than n / 2 is called?

A. Center
B. Diameter
C. Centroid
D. Path

52. What does the following code snippet do?

```
void dfs(int node, vector<vector<int>> &edges, vector<bool>
&vis, vector<int> &dp) {
   vis[node] = true;
   for(auto x: edges[node]) {
       if(!vis[x]) {
           dp[x] = dp[node] + 1;
           dfs(x, edges, vis, dp);
       }
   }
}
```

A. Stores depths of all the nodes in a given tree, with respect to some root node.
B. Counts the number of nodes in a given tree.
C. Finds the diameter of a tree.
D. Checks if all the nodes are reachable in a given tree.

53. Which of the following algorithms are used to find the shortest path from a source node to all other nodes in a weighted graph?

A. BFS.
B. Djikstra's Algorithm.
C. Prims Algorithm.
D. Kruskal's Algorithm.

54. What is the best time complexity we can achieve to precompute all-pairs shortest paths in a weighted graph?

A. O(n^3)
B. O(n^2)
C. O(n)
D. O(n^4)

55. Which data structure is mainly used for implementing the recursive algorithm?

A. Queue
B. Stack
C. Array
D. List