# https://www.javatpoint.com/data-structure-mcq

# Data Structure MCQ

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

1. The Array shows a hierarchical structure.
2. Arrays are immutable.
3. Container that stores the elements of similar types
4. The Array is not a data structure

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c because array stores the elements in a contiguous block of memory of similar types. Therefore, we can say that array is a container that stores the elements of similar types.

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

1. int javatpoint[10];
2. int javatpoint;
3. javatpoint{20};
4. array javatpoint[10];

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a because int specifies the type of the array, javatpoint is the name of the array, and 10 is the size of the array enclosed within the square brackets.

3) How can we initialize an array in C language?

1. int arr[2]=(10, 20)
2. int arr(2)={10, 20}
3. int arr[2] = {10, 20}
4. int arr(2) = (10, 20)

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c because the values assigned to the array must be enclosed within a curly bracket.

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

1. Elements of mixed data types can be stored.
2. Easier to access the elements in an array
3. Index of the first element starts from 1.
4. Elements of an array cannot be sorted

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b because the elements in an array are stored in a contiguous block of memory, so it is easier to access the elements of an array through indexing.

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

1. Binary Search tree
2. Caching
3. Spatial locality
4. Scheduling of Processes

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c, i.e., Spatial locality. Here, spatial locality means that the instruction accessed recently, then the nearby memory location would be accessed in the next iteration. As we know that in an array, all the elements are stored in a contiguous block of memory, so spatial locality is accessed quickly.

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

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

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c. For example, if we have an array of size 10 elements and we have inserted only 5 elements in an array then there is a wastage of 5 memory blocks which cannot be utilized by another variable.

7) What is the output of the below code?

1. #include <stdio.h>
2. **int** main()
3. {
4. **int** arr[5]={10,20,30,40,50};
5. printf("%d", arr[5]);
7. **return** 0;
8. }
9. Garbage value
10. 10
11. 50
12. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a because the indexing in an array starts from 0, so it starts from arr[0] to arr[4]. If we try to access arr[5] then the garbage value will be printed.

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

1. 9
2. 36
3. 35
4. None of the above

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b because the size of int type data is 4 bytes. The array stores 9 elements, so the size of the array is 9\*4=36 bytes.

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

1. Insert
2. Add
3. Push
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c. In stack, the process of inserting an element is known as a push operation.

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

1. Underflow
2. Garbage collection
3. Overflow
4. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a. Underflow is a condition that occurs when user tries to implement the pop operation in the empty stack.

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\_\_\_

1. Underflow
2. Garbage collection
3. Overflow
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c because the stack is full with its 10 elements, and inserting one more element in a stack will lead to the stack overflow.

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

1. String reversal
2. Recursion
3. Backtracking
4. Asynchronous data transfer

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. The first three options are the stack applications, but option d is not a stack application. The queue data structure is used for synchronization between the processes.

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

1. Queue
2. Stack
3. Binary tree
4. Linked list

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b. Recursion means calling the function itself again. Stack is used to maintain the previous records of the function.

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

1. Stack
2. Linked list
3. Binary tree
4. Queue

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., stack. Stack is a data structure used to reverse the order of the operators in the expression. It is also used as a storage structure that stores all the operators and print all the operators when all the operands have appeared.

15) Which of the following is the infix expression?

1. A+B\*C
2. +A\*BC
3. ABC+\*
4. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., A+B\*C because, in infix notation, all the operators appear between the operands.

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

1. A+(BC\*)
2. +AB\*C
3. ABC+\*
4. +A\*BC

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. The prefix notation means all the operators that appear before operand.

To convert the infix expression into a prefix expression, we will move the operator to the left of the parenthesis as shown in the below figure.

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

1. Arrays can be used to implement the stack
2. Stack follows FIFO
3. Elements are stored in a sequential manner
4. Top of the stack contains the last inserted element

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b because Stack does not follow FIFO. It follows LIFO.

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

1. 1234
2. 2134
3. 4321
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c because stack follows LIFO, which means that the element inserted at the last will be removed first.

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

1. 12
2. 11
3. 5
4. 4

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** Reverse of the prefix expression: 1, 4, 8, /, 2, 3, \*, -, +

| **Reading of prefix** | **Top of the stack** | **Stack representation** |
| --- | --- | --- |
| 1 | 1 | Data Structure MCQ |
| 4 | 4 | Data Structure MCQ |
| 8 | 8 | Data Structure MCQ |
| / | (8/4) | Data Structure MCQ |
| 2 | 2 | Data Structure MCQ |
| 3 | 3 | Data Structure MCQ |
| \* | (2\*3) | Data Structure MCQ |
| - | (2\*3) - (8/4) | Data Structure MCQ |
| + | (2\*3) - (8/4) +1 | Data Structure MCQ |

The infix expression of the above prefix expression is:

(2\*3) - (8/4) +1

6 -2 +1 = 5

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

1. 1
2. 3
3. 2
4. 5

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is 2. In Queue, one stack is required for the enqueue operation, and another stack will be used for the dequeue operation. The first stack is considered as the input stack whereas the second stack is considered as the output stack.

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

1. First node
2. Second node
3. Last node
4. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., First node. As we know, that last inserted element in the stack is considered as the top of the stack. Whenever the element is added to the linked list, it is always added at the beginning of the list. Therefore, we can say that the first node in the linked list is considered as the top of the stack.

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?

1. 8
2. 9
3. 11
4. 10

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is 10. The maximum size of the array is 11; therefore, we can insert 11 elements in the stack. The top value is initialized by -1, and on every insertion, the top value gets incremented.

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

1. Square buffer
2. Rectangle buffer
3. Ring buffer
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The circular queue is also known as a ring buffer. In a circular queue, the last element is connected back to the first element of the queue that forms a circle. Therefore, the structure of a circular queue is also known as a ring structure.

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

1. 1234
2. 4321
3. 3241
4. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., 1234. The queue follows the FIFO principle in which the element inserted first will be removed first.

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

1. Binary tree
2. Stack
3. Queue
4. Linked list

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is Queue. Queue is a data structure in which insertion takes place from one end, and deletion takes place from one end.

26) Which of the following principle does Queue use?

1. LIFO principle
2. FIFO principle
3. Linear tree
4. Ordered array

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is FIFO principle. Here, FIFO stands for First-In-First-Out. It means that the element which is inserted first will also be removed first.

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

1. Linear Queue
2. Circular Queue
3. Double ended Queue
4. Single ended Queue

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. i.e., single ended queue. Queue has two ends in which one end is used for the insertion and another end is used for the deletion. Therefore, it is not possible for the Queue to have a 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?

1. rear = front
2. rear = front+1
3. rear=MAX\_SIZE -1
4. rear = MAX\_SIZE

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c, i.e., rear=MAX\_SIZE-1. As the size of the array is MAX\_SIZE, so we can insert the elements till MAX\_SIZE-1. If we try to insert the elements of size MAX\_SIZE or more than MAX\_SIZE in a queue, then it leads to the overflow condition.

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

1. rear= MAX-1
2. rear=MAX
3. front=(rear+1) mod max
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c, i.e., front=(rear+1) mod max. The overflow condition for the linear queue is rear =MAX-1 as there is no space left in the Queue if rear = MAX-1. On the other hand, in a circular queue, the overflow condition is front=(rear+1) mod max because the last element is connected to the first element in a circular queue.

30) The time complexity of enqueue operation in Queue is \_\_

1. O(1)
2. O(n)
3. O(logn)
4. O(nlogn)

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., O(1). In Queue, the insertion is performed at the rear end, which is directly accessible; therefore, it takes O(1) time to insert an element in a Queue.

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

1. Avoid wastage of memory
2. Access the Queue using priority
3. Follows the FIFO principle
4. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., Avoid wastage of memory. In a linear queue, there are chances of wastage of memory because if the rear is pointing to the last element whereas the front is pointing to the element other than the first element; it means that spaces allocated before the front are free, but it cannot be reused as rear cannot be incremented. In contrast, the last element is connected to the first element in a circular queue; if initial spaces are vacant, then rear can be incremented by using the statement (rear+1) mod max where max is the size of the array. Therefore, we conclude that the circular queue avoids wastage of memory.

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

1. rear =rear+1
2. (rear+1) % max
3. (rear % max) + 1
4. None of the above

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b. It ensures that the rear will have the value from 0 to max-1; if the rear end points to the last position, and we increment the rear end using (rear+1) % max, then rear will point to the first position in the queue.

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. }
15. }

Which operation does the above code perform?

1. Enqueue
2. Dequeue
3. Return the front element
4. Both b and c

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d because two operations are performed in the above code. The first one is returning the value of the front with the help of the statement n=q[front], and the second operation is dequeue (deleting an element) by using the statement front++.

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

1. At the middle position of the linked list
2. At the head position of the linked list
3. At the tail position of the linked list
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c. If the queue is implemented using linked list, then the new element will be inserted at the tail position of the linked list as Queue follows FIFO principle in which new element will always be added at the end of the Queue.

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

1. 3
2. 2
3. 1
4. 4

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b because one queue is required to perform the push operation while the other queue is used to perform the pop operation.

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

1. Resource shared between various systems
2. Data is transferred asynchronously
3. Load balancing
4. Balancing of symbols

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:**

The answer is d. The options a, b, and c are the applications of the Queue data structure while option d, i.e., balancing of symbols is not the application of the Queue data structure. The option a, i.e., resource shared between various system is the application of the Queue data structure as it allows to align all the requests for the resource in a queue. The option b, i.e., data is transferred asynchronously is a application of the Queue data structure. Here asynchronously means that the data received at the different rate as sent.

The option c, i.e., load balancing is also an application of the Queue data structure because all the requests from the client are stored in the Queue, and it distributes the requests to the client one by one.

The option d, i.e., balancing of symbols is an application of the stack data structure.

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

1. In enqueue operation, new nodes are inserted from the beginning and in dequeue operation, nodes are removed from the end.
2. In enqueue operation, new nodes are inserted from the end and in dequeue operation, nodes are deleted from the beginning.
3. In enqueue operation, new nodes are inserted from the end and in dequeue operation, nodes are deleted from the end.
4. Both a and b

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. As we know that Queue has two ends, i.e., one for the insertion and another one for the deletion. If Queue is implemented using Linked list then it can be done in either of the ways.

38) The necessary condition to be checked before deletion from the Queue is\_\_

1. Overflow
2. Underflow
3. Rear value
4. Front value

**Show Answer** **Workspace**

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

1. Stack
2. Linked list
3. Array
4. Heap

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d, i.e., Heap. All the data structures that are given in the above options can be used to implement a priority queue but the most efficient way of implementing a priority queue is heap data structure.

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

1. LIFO
2. FIFO
3. Linear tree
4. None of the above

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b, i.e., FIFO. In a priority queue, if two or more elements have the same priority then they are arranged based on the FIFO principle.

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

1. Processes with different priority can be easily handled
2. Easy to implement
3. Deletion is easier
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c. i.e., deletion is easier. In worst case, the deletion is not easier as we have to traverse to the n elements until the element to be removed is not found.

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

1. Queue
2. Deque
3. Priority queue
4. Circular queue

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b, i.e., Deque. The deque is a data structure in which both insertion and deletion can be performed from both the ends whereas, in Queue, insertion can be done from one end and deletion can be performed from another end.

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

1. O(1)
2. O(n2)
3. O(n)
4. O(nlogn)

**Hide Answer** **Workspace**

**Answer:** O(n)

**Explanation:** The answer is O(n) because we need to traverse till the n element to delete the element from the rear end.

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

1. Input-restricted queue
2. Output-restricted queue
3. Priority queue
4. None of the above

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b. The output-restricted queue is one of the types of the Deque data structure in which insertion is allowed from both the ends but the deletion is allowed from only one end.

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();
9. 10, 20, 30
10. 50, 10, 30
11. 40, 20, 30
12. None of the above

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:**

The answer is b.

Let's dry run the above code.

When insertfront(10) is called, deque would be:

10

When insertfront(20) is called, the deque would be:

20 10

When insertrear(30) is called, the deque would be:

20 10 30

When insertrear(40) is called, the deque would be:

20 10 30 40

When deletefront() is called, the deque would be:

10 30 40

When insertfront(50) is called, the deque would be:

50 10 30 40

When deleterear() is called, the deque would be:

50 10 30

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.

1. 5
2. 0
3. 1
4. 2

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b, i.e., 0. As it is mentioned in the question that the size of the array is 5; therefore, the range would be from 0 to 4. In a circular queue, the last element is connected to the first element; the value of rear is 4 so when we increment the value then it will point to the 0th position of the array.

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?

1. Front=rear= -1
2. Front=rear=0
3. Front=rear+1
4. None of the above

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a, i.e., front=rear= -1. When the circular queue is empty means that no element is available in the queue then the front and rear are initialized with a -1 value.

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

1. ii
2. both ii and iii
3. both i and iv
4. both i and ii

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b. As it is mentioned in the above question that there is only head pointer pointing to the front node in the list so it will take O(1) time to insert at the front as well as to delete the first node from the list. If we try to insert the node at the end or delete the last node then it will take O(n) time as we need to traverse till the n elements.

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)?

1. O(1)
2. O(n)
3. O(logn)
4. O(nlogn)

**Hide Answer** **Workspace**

**Answer:** b

**Explanation:** The answer is b, i.e., O(n). As it is mentioned in the above question that head pointer is known, so to insert the node at the end of the linked list; we have to traverse till the nth node. Therefore, the time complexity would be O(n).

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

1. O(1)
2. O(n)
3. O(logn)
4. O(nlogn)

**Hide Answer** **Workspace**

**Answer:** O(n)

**Explanation:** The answer is O(n). The worst-case time complexity to search an element in the linked list is O(n) because if we have to find the last element then we need to traverse till the nth node.

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?

1. ptr= (node\*)malloc(sizeof(node\*))
2. ptr=(node)malloc(sizeof(node))
3. ptr=(node\*)malloc(sizeof(node))
4. None of the above

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c, i.e., ptr=(node\*)malloc(sizeof(node)). In this statement, we have used a malloc() function for allocating the memory to the node and ptr is a pointer variable that will point to the newly created node.

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

1. We can traverse in both the directions.
2. It requires extra space
3. Implementation of doubly linked list is easier than the singly linked list
4. It stores the addresses of the next and the previous node

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c. The implementation of doubly linked list is complex as compared to singly linked list as it needs to store the addresses of the two nodes, i.e., the previous and the next node. If we insert or delete the node from the doubly linked list then it needs to adjust two pointers, previous and the next pointer.

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

1. 3
2. 1
3. 4
4. 2

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. The binary tree can contain utmost two children.

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

1. Randomized traversal
2. Preorder traversal
3. Postorder traversal
4. Inorder traversal

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a. The binary tree contains three traversal techniques, preorder, postorder and inorder traversal.

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

1. The value of the left child should be less than the root node
2. The value of the right child should be greater than the root node.
3. The left and right sub trees should also be a binary search tree
4. None of the above

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. All the three options, i.e., a, b and c are true for the binary search tree. In binary search tree, the left child should be less than the root node and the right child should be greater than the value of the root node.

56) How can we define a AVL tree?

1. A tree which is binary search tree and height balanced tree.
2. A tree which is a binary search tree but unbalanced tree.
3. A tree with utmost two children
4. A tree with utmost three children

**Hide Answer** **Workspace**

**Answer:** a

**Explanation:** The answer is a. An AVL tree is a binary search tree and height balanced tree.

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

1. Red Black trees are not strictly balanced
2. Red black tree requires lesser rotations than AVL tree.
3. AVL tree needs more space to store the balance factor.
4. Both b and c

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. Red Black tree requires fewer rotations for inserting or deleting the node whereas AVL tree requires multiple rotations to balance the tree. AVL tree stores balance factor per node that requires space whereas Red Black tree stores 1-bit information.

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

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

**Hide Answer** **Workspace**

**Answer:** d

**Explanation:** The answer is d. Red black tree is a binary search tree but it is not a strictly balanced tree like AVL tree. In Red Black tree, a node must be either in Black or Red in color and root node must be in Black color.

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

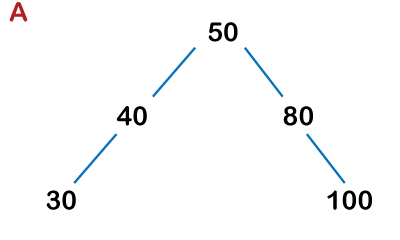
1. Black, if the new node is not a root node
2. Red, if the new node is not a root node
3. Black, if the new node is a root node
4. Both b and c

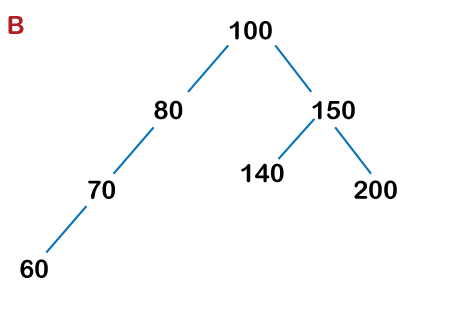
**Hide Answer** **Workspace**

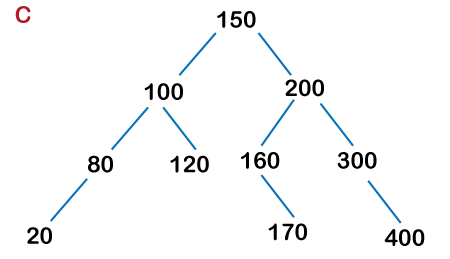
**Answer:** d

**Explanation:** The answer is d. The property of Red Black tree is that if the newly created node is a root node then the color of the node would be Black otherwise the color of the node would be Red.

60) Identify the AVL tree among the following options?







1. A
2. C
3. Both A and C
4. B

**Hide Answer** **Workspace**

**Answer:** c

**Explanation:** The answer is c. Both A and C are the AVL trees but B is not a AVL tree. As we know that balance factor of every node in AVL tree should be either -1, 0 or 1. But in case of B, the balance factor of node 80 is 2, so it is not a AVL tree.