# Data Structure Questions and Answers – Stack Operations – 1

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack Operations – 1”.

1. Process of inserting an element in stack is called _____
a) Create
b) Push
c) Evaluation
d) Pop
View Answer
Answer: b
Explanation: Push operation allows users to insert elements in the stack. If the stack is filled completely and trying to perform push operation stack – overflow can happen.

2. Process of removing an element from stack is called _____
a) Create
b) Push
c) Evaluation
d) Pop
View Answer
Answer: d
Explanation: Elements in the stack are removed using pop operation. Pop operation removes the top most element in the stack i.e. last entered element.

3. 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
View Answer
Answer: a
Explanation: Underflow occurs when the user performs a pop operation on an empty stack. Overflow occurs when the stack is full and the user performs a push operation. Garbage Collection is used to recover the memory occupied by objects that are no longer used.

4. 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
View Answer
Answer: a
Explanation: The stack is filled with 5 elements and pushing one more element causes a stack overflow. This results in overwriting memory, code and loss of unsaved work on the computer.

5. 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
View Answer
Answer: d
Explanation: In stack data structure, elements are added one by one using push operation. Stack follows LIFO Principle i.e. Last In First Out(LIFO).



6. 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
View Answer
Answer: d
Explanation: Data transfer between the two asynchronous process uses the queue data structure for synchronisation. The rest are all stack applications.

7. 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
View Answer
Answer: c
Explanation: In the entire parenthesis balancing method when the incoming token is a left parenthesis it is pushed into stack. A right parenthesis makes pop operation to delete the elements in stack till we get left parenthesis as top most element. 3 elements are there in stack before right parentheses comes. Therefore, maximum number of elements in stack at run time is 3.

8. 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
View Answer
Answer: b
Explanation: In the entire parenthesis balancing method when the incoming token is a left parenthesis it is pushed into stack. A right parenthesis makes pop operation to delete the elements in stack till we get left parenthesis as top most element. 2 left parenthesis are pushed whereas one right parenthesis removes one of left parenthesis. 2 elements are there before right parenthesis which is the maximum number of elements in stack at run time.

9. What is the value of the postfix expression 6 3 2 4 + – *?
a) 1
b) 40
c) 74
d) -18
View Answer
Answer: d
Explanation: Postfix Expression is (6*(3-(2+4))) which results -18 as output.

10. 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
View Answer
Answer: d
Explanation: When we perform the conversion from infix to postfix expression +, *, (, * symbols are placed inside the stack. A maximum of 4 symbols are identified during the entire conversion.

# Data Structure Questions and Answers – Stack Operations – 2

This set of Data Structure Interview Questions and Answers focuses on “Stack Operations – 2”.

1. 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 /
View Answer
Answer: c
Explanation: (((A+ B)*(C*D- E)*F) / G) is converted to postfix expression as
(AB+(*(C*D- E)*F )/ G)
(AB+CD*E-*F) / G
(AB+CD*E-*F * G/). Thus Postfix expression is AB+CD*E-*F*G/
2. The data structure required to check whether an expression contains a balanced parenthesis is?
a) Stack
b) Queue
c) Array
d) Tree
View Answer
Answer: a
Explanation: The stack is a simple data structure in which elements are added and removed based on the LIFO principle. Open parenthesis is pushed into the stack and a closed parenthesis pops out elements till the top element of the stack is its corresponding open parenthesis. If the stack is empty, parenthesis is balanced otherwise it is unbalanced.
3. 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
View Answer
Answer: b
Explanation: In recursive algorithms, the order in which the recursive process comes back is the reverse of the order in which it goes forward during execution. The compiler uses the stack data structure to implement recursion. In the forwarding phase, the values of local variables, parameters and the return address are pushed into the stack at each recursion level. In the backing-out phase, the stacked address is popped and used to execute the rest of the code.
4. 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
View Answer
Answer: d
Explanation: In serial access memory data records are stored one after the other in which they are created and are accessed sequentially. In stack data structure, elements are accessed sequentially. Stack data structure resembles the serial access memory.
5. The postfix form of A*B+C/D is?
a) *AB/CD+
b) AB*CD/+
c) A*BC+/D
d) ABCD+/*
View Answer
Answer: b
Explanation: Infix expression is (A*B)+(C/D)
AB*+(C/D)
AB*CD/+. Thus postfix expression is AB*CD/+.

6. Which data structure is needed to convert infix notation to postfix notation?
a) Branch
b) Tree
c) Queue
d) Stack
View Answer
Answer: d
Explanation: The Stack data structure is used to convert infix expression to postfix expression. The purpose of stack is to reverse the order of the operators in the expression. It also serves as a storage structure, as no operator can be printed until both of its operands have appeared.
7. The prefix form of A-B/ (C * D ^ E) is?
a) -/*^ACBDE
b) -ABCD*^DE
c) -A/B*C^DE
d) -A/BC*^DE
View Answer
Answer: c
Explanation: Infix Expression is (A-B)/(C*D^E)
(-A/B)(C*D^E)
-A/B*C^DE. Thus prefix expression is -A/B*C^DE.
8. What is the result of the following operation?
**Top (Push (S, X))**
a) X
b) X+S
c) S
d) XS
View Answer
Answer: a
Explanation: The function Push(S,X) pushes the value X in the stack S. Top() function gives the value which entered last. X entered into stack S at last.
9. The prefix form of an infix expression (p + q) – (r * t) is?
a) + pq – *rt
b) – +pqr * t
c) – +pq * rt
d) – + * pqrt
View Answer
Answer: c
Explanation: Given Infix Expression is ((p+q)-(r*t))
(+pq)-(r*t)
(-+pq)(r*t)
-+pq*rt. Thus prefix expression is -+pq*rt.
10. Which data structure is used for implementing recursion?
a) Queue
b) Stack
c) Array
d) List
View Answer
Answer: b
Explanation: Stacks are used for the implementation of Recursion.

## Data Structure Questions and Answers – Queue Operations

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Queue Operations”.

1. 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
View Answer
Answer: a
Explanation: Linear list of elements in which deletion is done at front side and insertion at rear side is called Queue. In stack we will delete the last entered element first.

2. The data structure required for Breadth First Traversal on a graph is?
a) Stack
b) Array
c) Queue
d) Tree
View Answer
Answer: c
Explanation: In Breadth First Search Traversal, BFS, starting vertex is first taken and adjacent vertices which are unvisited are also taken. Again, the first vertex which was added as an unvisited adjacent vertex list will be considered to add further unvisited vertices of the graph. To get the first unvisited vertex we need to follows First In First Out principle. Queue uses FIFO principle.

3. A queue follows _____
a) FIFO (First In First Out) principle
b) LIFO (Last In First Out) principle
c) Ordered array
d) Linear tree
View Answer
Answer: a
Explanation: Element first added in queue will be deleted first which is FIFO principle.

4. Circular Queue is also known as _____
a) Ring Buffer
b) Square Buffer
c) Rectangle Buffer
d) Curve Buffer
View Answer
Answer: a
Explanation: Circular Queue is also called as Ring Buffer. Circular Queue is a linear data structure in which last position is connected back to the first position to make a circle. It forms a ring structure.



5. 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
View Answer
Answer: a
Explanation: Queue follows FIFO approach. i.e. First in First Out Approach. So, the order of removal elements are ABCD.

6. 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
View Answer
Answer: c
Explanation: In dequeuer, we can insert or delete elements from both the ends. In queue, we will follow first in first out principle for insertion and deletion of elements. Element with least priority will be deleted in a priority queue.

7. 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
View Answer
Answer: a
Explanation: When Rear = MAX_SIZE – 1, there will be no space left for the elements to be added in queue. Thus queue becomes full.

8. 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
View Answer
Answer: c
Explanation: Simulation of recursion uses stack data structure. Simulation of arbitrary linked lists uses linked lists. Simulation of resource allocation uses queue as first entered data needs to be given first priority during resource allocation. Simulation of heap sort uses heap data structure.

9. Which of the following is not the type of queue?
a) Ordinary queue
b) Single ended queue
c) Circular queue
d) Priority queue
View Answer
Answer: b
Explanation: Queue always has two ends. So, single ended queue is not the type of queue.

## Data Structure Questions and Answers – Singly Linked List Operations – 1

This set of Data Structure Interview Questions & Answers focuses on “Singly Linked List Operations – 1”.

1. 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
View Answer
Answer: a
Explanation: In Linked list each node has its own data and the address of next node. These nodes are linked by using pointers. Node list is an object that consists of a list of all nodes in a document with in a particular selected set of nodes.

2. 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
View Answer
Answer: b
Explanation: We know the head node in the given linked list. Insertion and deletion of elements at the front of the linked list completes in O (1) time whereas for insertion and deletion at the last node requires to traverse through every node in the linked list. Suppose there are n elements in a linked list, we need to traverse through each node. Hence time complexity becomes O(n).

3. 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
View Answer
Answer: c
Explanation: Each node in a linked list contains data and a pointer (reference) to the next node. Second field contains pointer to node.



4. 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)
View Answer
Answer: c
Explanation: In case of a linked list having n elements, we need to travel through every node of the list to add the element at the end of the list. Thus asymptotic time complexity is θ(n).

5. 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) $O(n^3)$
View Answer
Answer: a
Explanation: To add an element at the front of the linked list, we will create a new node which holds the data to be added to the linked list and pointer which points to head position in the linked list. The entire thing happens within O (1) time. Thus the asymptotic time complexity is O (1).

6. 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) $O(n^4)$
View Answer
Answer: b
Explanation: If the required element is in the last position, we need to traverse the entire linked list. This will take O (n) time to search the element.

7. 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) $O(n^3)$
View Answer
Answer: a
Explanation: A new node is created with the required element. The pointer of the new node points the node to which the head node of the linked list is also pointing. The head node pointer is changed and it points to the new node which we created earlier. The entire process completes in O (1) time. Thus the asymptotic time complexity to insert an element in the second position of the linked list is O (1).

8. 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
View Answer
Answer: c
Explanation: We can easily concatenate two lists in O (1) time using singly or doubly linked list, provided that we have a pointer to the last node at least one of the lists. But in case of circular doubly linked lists, we will break the link in both the lists and hook them together. Thus circular doubly linked list concatenates two lists in O (1) time.

9. 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));
View Answer
Answer: a
Explanation: As it represents the right way to create a node.

## Data Structure Questions and Answers – Singly Linked List Operations – 2

This set of Data Structure Interview Questions and Answers for freshers focuses on “Singly Linked Lists Operations – 2”.

1. 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
View Answer
Answer: d
Explanation: Arrays provide random access to elements by providing the index value within square brackets. In the linked list, we need to traverse through each element until we reach the nth position. Time taken to access an element represented in arrays is less than the singly, doubly and circular linked lists. Thus, array implementation is used to access the item at the position n.

2. Linked lists are not suitable for the implementation of _____
a) Insertion sort
b) Radix sort
c) Polynomial manipulation
d) Binary search
View Answer
Answer: d
Explanation: It cannot be implemented using linked lists.

3. Linked list is considered as an example of _____ type of memory allocation.
a) Dynamic
b) Static
c) Compile time
d) Heap
View Answer
Answer: a
Explanation: As memory is allocated at the run time.

4. In Linked List implementation, a node carries information regarding _____
a) Data
b) Link
c) Data and Link
d) Node
View Answer
Answer: b
Explanation: A linked list is a collection of objects linked together by references from an object to another object. By convention these objects are names as nodes. Linked list consists of nodes where each node contains one or more data fields and a reference(link) to the next node.

5. Linked list data structure offers considerable saving in _____
a) Computational Time
b) Space Utilization
c) Space Utilization and Computational Time
d) Speed Utilization
View Answer
Answer: c
Explanation: Linked lists saves both space and time.



6. 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
View Answer
Answer: d
Explanation: To access an element in a linked list, we need to traverse every element until we reach the desired element. This will take more time than arrays as arrays provide random access to its elements.

7. 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
View Answer
Answer: b
Explanation: fun1() prints the given Linked List in reverse manner.
For Linked List 1->2->3->4->5, fun1() prints 5->4->3->2->1.

8. 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
View Answer
Answer: d
Explanation: Both Merge sort and Insertion sort can be used for linked lists. The slow random-access performance of a linked list makes other algorithms (such as quicksort) perform poorly, and others (such as heapsort) completely impossible. Since worst case time complexity of Merge Sort is O(nLogn) and Insertion sort is $O(n^2)$, merge sort is preferred.

## Infix to Prefix Conversion Multiple Choice Questions and Answers (MCQs)

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Infix to Prefix Conversion”.

1. What data structure is used when converting an infix notation to prefix notation?
a) Stack
b) Queue
c) B-Trees

d) Linked-list
View Answer
Answer: a
Explanation: First you reverse the given equation and carry out the algorithm of infix to postfix expression. Here, the data structure used is stacks.
2. 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
View Answer
Answer: c
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-postfix algorithm. The equation inside the bracket evaluates to CB* and outside the bracket evaluates to A+ therefore getting CB*A+. Reversing this and we get +A*BC.
3. 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
View Answer
Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-postfix algorithm. The equation inside the brackets evaluate to DC* and BA* respectively giving us DC*BA*+ in the end. Reversing this we get the +*AB*CD.

4. 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
View Answer
Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. The preference order in ascending order are as follows +*^. Operators are pushed into the stack and popped if its preference is greater than the one which is getting pushed. In the end all operators are popped. The equation evaluates to DC^B*A+. Reversing this we get our following answer.
5. Out of the following operators (^, *, +, &, $), the one having highest priority is _____
a) +
b) $
c) ^
d) &
View Answer
Answer: c
Explanation: According to the algorithm (infix-prefix), it follows that the exponentiation will have the highest priority.
6. Out of the following operators (|, *, +, &, $), the one having lowest priority is _____
a) +
b) $
c) |
d) &
View Answer
Answer: c
Explanation: According to the algorithm (infix-prefix), it follows that the logical OR will have the lowest priority.
7. 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
View Answer
Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. Here we have to remember that the exponentiation has order of associativity from right to left. Therefore the stack goes on pushing ^. Therefore resulting in ^^^ABCD.
8. 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
View Answer
Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. The preference order in ascending order are as follows |&+*/.
9. 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
View Answer
Answer: b
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. The preference order in ascending order are as follows +*/^. Brackets have the highest priority. The equations inside the brackets are solved first.
10. 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+
View Answer

Answer: a
Explanation: For prefix notation there is a need of reversing the giving equation and solving it as a normal infix-postfix question. We see that it doesn't result as same as normal infix-postfix conversion.
11. What would be the Prefix notation for the given equation?

```
a|b&c
```

a) a|&bc
b) &|abc
c) |a&bc
d) ab&|c
View Answer
Answer: c
Explanation: The order of preference of operators is as follows (descending): & |.
The equation **a|b&c** will be parenthesized as **(a|(b&c))** for evaluation.
Therefore the equation for prefix notation evaluates to **|a&bc**.

## Infix to Postfix Conversion Multiple Choice Questions and Answers (MCQs)

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Infix to Postfix Conversion".
1. 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
View Answer
Answer: a
Explanation: While converting an infix expression to a postfix expression, when an operand is read, it is placed on to the output. When an operator is read, it is placed in the operator stack.
2. 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
View Answer
Answer: c
Explanation: When a left parenthesis is encountered, it is placed on to the operator stack. When the corresponding right parenthesis is encountered, the stack is popped until the left parenthesis and remove both the parenthesis.
3. Which of the following is an infix expression?
a) (a+b)*(c+d)
b) ab+c*
c) +ab
d) abc+*
View Answer
Answer: a
Explanation: (a+b)*(c+d) is an infix expression. +ab is a prefix expression and ab+c* is a postfix expression.
4. 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)
View Answer
Answer: b
Explanation: The time complexity of an infix to postfix expression conversion algorithm is mathematically found to be O(N).
5.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)*+
View Answer
Answer: a
Explanation: Using the infix to postfix expression conversion algorithm, the corresponding postfix expression is found to be abc*+de*+.

6. Parentheses are simply ignored in the conversion of infix to postfix expression.
a) True
b) False
View Answer
Answer: b
Explanation: When a parenthesis is encountered, it is placed on the operator stack. When the corresponding parenthesis is encountered, the stack is popped until the other parenthesis is reached and they are discarded.
7. It is easier for a computer to process a postfix expression than an infix expression.
a) True
b) False
View Answer
Answer: a
Explanation: Computers can easily process a postfix expression because a postfix expression keeps track of precedence of operators.
8. What is the postfix expression for the infix expression?

```
a-b-c
```

a) -ab-c
b) ab – c –
c) – -abc
d) -ab-c
View Answer
Answer: b
Explanation: The corresponding postfix expression for the given infix expression is found to be ab-c- and not abc- -.
9. 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^/-
View Answer

Answer: a
Explanation: Using the infix to postfix conversion algorithm, the corresponding postfix expression for the infix expression is found to be abc^/d-.
10. 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
View Answer
Answer: c
Explanation: Parentheses are not included in the output. They are placed in the operator stack and then discarded.
11. 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
View Answer
Answer: b
Explanation: In infix, prefix and postfix expressions, the operators are associated from left to right and not right to left.
12. What is the corresponding postfix expression for the given infix expression?

```
a*(b+c)/d
```

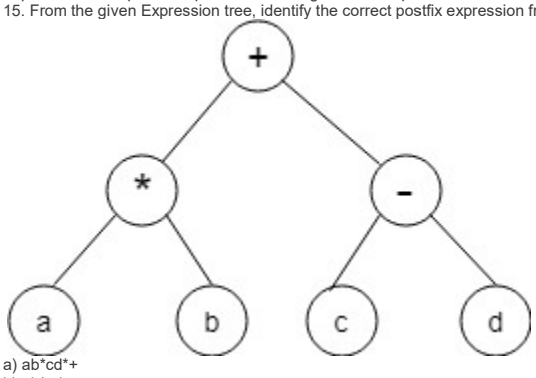
a) ab*+cd/
b) ab+*cd/
c) abc*+/d
d) abc+*d/
View Answer
Answer: d
Explanation: Using the infix to postfix conversion algorithm, the corresponding postfix expression is obtained as abc+*d/.
13. What is the corresponding postfix expression for the given infix expression?

```
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+
View Answer
Answer: b
Explanation: Using the infix to postfix expression conversion algorithm using stack, the corresponding postfix expression is found to be abcdef^/*g*h*+.
14. 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-
View Answer
Answer: c
Explanation: The postfix expression for the given infix expression is found to be abcd^e-fgh*+^*+i- when we use infix to postfix conversion algorithm.
15. 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-
View Answer
Answer: b
Explanation: From the given expression tree, the infix expression is found to be (a*b)+(c-d). Converting it to postfix, we get, ab*cd-+.

## Data Structure Questions and Answers – Queue using Stacks

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Queue using Stacks”.
1. A Double-ended queue supports operations such as adding and removing items from both the sides of the queue. They support four operations like addFront(adding item to top of the queue), addRear(adding item to the bottom of the queue), removeFront(removing item from the top of the queue) and removeRear(removing item from the bottom of the queue). You are given only stacks to implement this data structure. You can implement only push and pop operations. What are the total number of stacks required for this operation?(you can reuse the stack)
a) 1
b) 2
c) 3
d) 4
View Answer
Answer: b
Explanation: The addFront and removeFront operations can be performed using one stack itself as push and pop are supported (adding and removing element from top of the stack) but to perform addRear and removeRear you need to pop each element from the current stack and push it into another stack, push or pop the element as per the asked operation from this stack and in the end pop elements from this stack to the first stack.
2. You are asked to perform a queue operation using a stack. Assume the size of the stack is some value ‘n’ and there are ‘m’ number of variables in this stack. The time complexity of performing deQueue operation is (Using only stack operations like push and pop)(Tightly bound).
a) O(m)

b) O(n)
c) O(m*n)
d) Data is insufficient
View Answer
Answer: a
Explanation: To perform deQueue operation you need to pop each element from the first stack and push it into the second stack. In this case you need to pop 'm' times and need to perform push operations also 'm' times. Then you pop the first element from this second stack (constant time) and pass all the elements to the first stack (as done in the beginning)('m-1' times). Therfore the time complexity is O(m).
3. Consider you have an array of some random size. You need to perform dequeue operation. You can perform it using stack operation (push and pop) or using queue operations itself (enQueue and Dequeue). The output is guaranteed to be same. Find some differences?
a) They will have different time complexities
b) The memory used will not be different
c) There are chances that output might be different
d) No differences
View Answer
Answer: a
Explanation: To perform operations such as Dequeue using stack operation you need to empty all the elements from the current stack and push it into the next stack, resulting in a O(number of elements) complexity whereas the time complexity of dequeue operation itself is O(1). And there is a need of a extra stack. Therefore more memory is needed.
4. Consider you have a stack whose elements in it are as follows.
5 4 3 2 << top
Where the top element is 2.
You need to get the following stack
6 5 4 3 2 << top
The operations that needed to be performed are (You can perform only push and pop):
a) Push(pop()), push(6), push(pop())
b) Push(pop()), push(6)
c) Push(pop()), push(pop()), push(6)
d) Push(6)
View Answer
Answer: a
Explanation: By performing push(pop()) on all elements on the current stack to the next stack you get 2 3 4 5 << top.Push(6) and perform push(pop()) you'll get back 6 5 4 3 2 << top. You have actually performed enQueue operation using push and pop.
5. A double-ended queue supports operations like adding and removing items from both the sides of the queue. They support four operations like addFront(adding item to top of the queue), addRear(adding item to the bottom of the queue), removeFront(removing item from the top of the queue) and removeRear(removing item from the bottom of the queue). You are given only stacks to implement this data structure. You can implement only push and pop operations. What's the time complexity of performing addFront and addRear? (Assume 'm' to be the size of the stack and 'n' to be the number of elements)
a) O(m) and O(n)
b) O(1) and O(n)
c) O(n) and O(1)
d) O(n) and O(m)
View Answer
Answer: b
Explanation: addFront is just a normal push operation. Push operation is of O(1). Whereas addRear is of O(n) as it requires two push(pop()) operations of all elements of a stack.

6. Why is implementation of stack operations on queues not feasible for a large dataset (Asssume the number of elements in the stack to be n)?
a) Because of its time complexity O(n)
b) Because of its time complexity O(log(n))
c) Extra memory is not required
d) There are no problems
View Answer
Answer: a
Explanation: To perform Queue operations such as enQueue and deQueue there is a need of emptying all the elements of a current stack and pushing elements into the next stack and vice versa. Therfore it has a time complexity of O(n) and the need of extra stack as well, may not be feasible for a large dataset.
7. Consider yourself to be in a planet where the computational power of chips to be slow. You have an array of size 10.You want to perform enqueue some element into this array. But you can perform only push and pop operations .Push and pop operation both take 1 sec respectively. The total time required to perform enQueue operation is?
a) 20
b) 40
c) 42
d) 43
View Answer
Answer: d
Explanation: First you have to empty all the elements of the current stack into the temporary stack, push the required element and empty the elements of the temporary stack into the original stack. Therfore taking 10+10+1+11+11= 43 seconds.
8. You have two jars, one jar which has 10 rings and the other has none. They are placed one above the other. You want to remove the last ring in the jar. And the second jar is weak and cannot be used to store rings for a long time.
a) Empty the first jar by removing it one by one from the first jar and placing it into the second jar
b) Empty the first jar by removing it one by one from the first jar and placing it into the second jar and empty the second jar by placing all the rings into the first jar one by one
c) There exists no possible way to do this
d) Break the jar and remove the last one
View Answer
Answer: b
Explanation: This is similar to performing dequeue operation using push and pop only. Elements in the first jar are taken out and placed in the second jar. After removing the last element from the first jar, remove all the elements in the second jar and place them in the first jar.
9. Given only a single array of size 10 and no other memory is available. Which of the following operation is not feasible to implement (Given only push and pop operation)?
a) Push
b) Pop
c) Enqueue
d) Returntop
View Answer
Answer: c
Explanation: To perform Enqueue using just push and pop operations, there is a need of another array of same size. But as there is no extra available memeory, the given operation is not feasible.
10. Given an array of size n, let's assume an element is 'touched' if and only if some operation is performed on it(for example, for performing a pop operation the top element is 'touched'). Now you need to perform Dequeue operation. Each element in the array is touched atleast?
a) Once
b) Twice
c) Thrice
d) Four times
View Answer
Answer: d
Explanation: First each element from the first stack is popped, then pushed into the second stack, dequeue operation is done on the top of the stack and later the each element of second stack is popped then pushed into the first stack. Therfore each element is touched four times.

# B-Tree Multiple Choice Questions and Answers (MCQs)

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “B-Tree”.

1. Which of the following is the most widely used external memory data structure?
a) AVL tree
b) B-tree
c) Red-black tree
d) Both AVL tree and Red-black tree
View Answer
Answer: b
Explanation: In external memory, the data is transferred in form of blocks. These blocks have data valued and pointers. And B-tree can hold both the data values and pointers. So B-tree is used as an external memory data structure.

2. B-tree of order n is a order-n multiway tree in which each non-root node contains _____
a) at most (n – 1)/2 keys
b) exact (n – 1)/2 keys
c) at least 2n keys
d) at least (n – 1)/2 keys
View Answer
Answer: d
Explanation: A non-root node in a B-tree of order n contains at least (n – 1)/2 keys. And contains a maximum of (n – 1) keys and n sons.

3. A B-tree of order 4 and of height 3 will have a maximum of _____ keys.
a) 255
b) 63
c) 127
d) 188
View Answer
Answer: a
Explanation: A B-tree of order m of height h will have the maximum number of keys when all nodes are completely filled. So, the B-tree will have n = $(m^{h+1} – 1)$ keys in this situation. So, required number of maximum keys = $4^{3+1} – 1 = 256 – 1 = 255$.

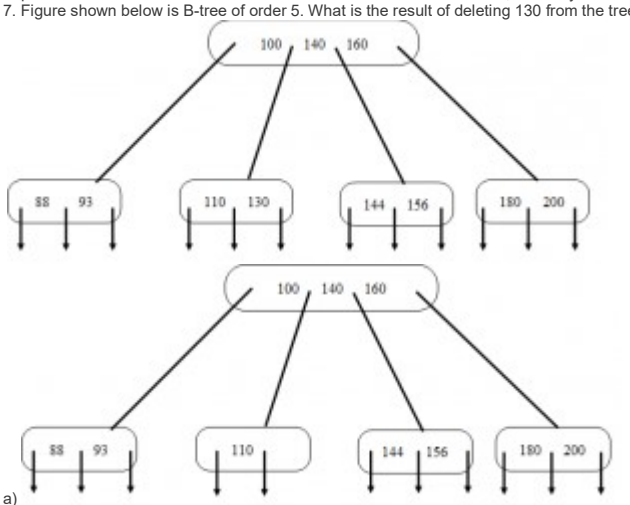
4. Five node splitting operations occurred when an entry is inserted into a B-tree. Then how many nodes are written?
a) 14
b) 7
c) 11
d) 5
View Answer
Answer: c
Explanation: If s splits occur in a B-tree, 2s + 1 nodes are written (2 halves of each split and the parent of the last node split). So, if 5 splits occurred, then 2 * 5 + 1, i.e. 11 nodes are written.

5. B-tree and AVL tree have the same worst case time complexity for insertion and deletion.
a) True
b) False
View Answer
Answer: a
Explanation: Both the B-tree and the AVL tree have O(log n) as worst case time complexity for insertion and deletion.

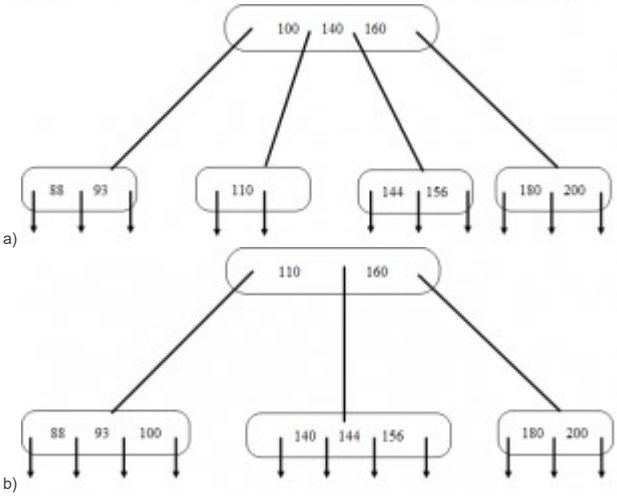


6. 2-3-4 trees are B-trees of order 4. They are an isometric of _____ trees.
a) AVL
b) AA
c) 2-3
d) Red-Black
View Answer
Answer: d
Explanation: 2-3-4 trees are isometric of Red-Black trees. It means that, for every 2-3-4 tree, there exists a Red-Black tree with data elements in the same order.

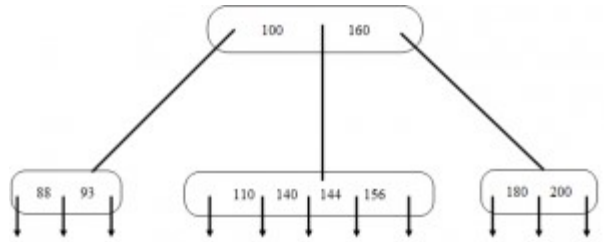
7. Figure shown below is B-tree of order 5. What is the result of deleting 130 from the tree?

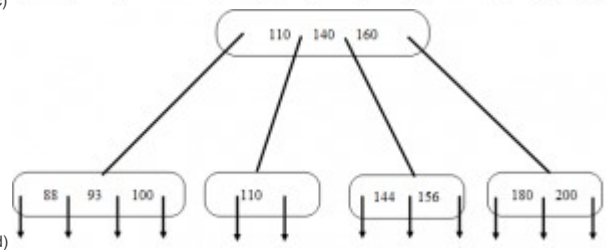



a)




b)

c)


d)


View Answer
Answer: c
Explanation: Each non-root in a B-tree of order 5 must contain at least 2 keys. Here, when the key 130 is deleted the node gets underflowed i.e. number of keys in the node drops below 2. So we combine the node with key 110 with it's brother node having keys 144 and 156. And this combined node will also contain the separator key from parent i.e. key 140, leaving the root with two keys 110 and 160.
8. What is the best case height of a B-tree of order n and which has k keys?
a) logn (k+1) – 1
b) nk
c) $\log_k (n+1) - 1$
d) $k\log_n$
View Answer
Answer: a
Explanation: B-tree of order n and with height k has best case height h, where h = logn (k+1) – 1. The best case occurs when all the nodes are completely filled with keys.
9. Compression techniques can be used on the keys to reduce both space and time requirements in a B-tree.
a) True
b) False
View Answer
Answer: a
Explanation: The front compression and the rear compression are techniques used to reduce space and time requirements in B-tree. The compression enables to retain more keys in a node so that the number of nodes needed can be reduced.
10. Which of the following is true?
a) larger the order of B-tree, less frequently the split occurs
b) larger the order of B-tree, more frequently the split occurs
c) smaller the order of B-tree, more frequently the split occurs
d) smaller the order of B-tree, less frequently the split occurs
View Answer
Answer: a

## Data Structure Questions and Answers – Binary Trees using Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Binary Trees using Array".
1. 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
View Answer
Answer: c
Explanation: Can have atmost 2 nodes.
2. 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
View Answer
Answer: c
Explanation: The size of array is fixed in normal arrays. We need to know the number of nodes in the tree before array declaration. It is the main disadvantage of using arrays to represent binary trees.
3. What must be the ideal size of array if the height of tree is 'l'?
a) 2l-1
b) l-1
c) l
d) 2l
View Answer
Answer: a
Explanation: Maximum elements in a tree (complete binary tree in worst case) of height 'L' is $2^L$-1. Hence size of array is taken as $2^L$-1.
4. 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
View Answer
Answer: a
Explanation: The left child is generally taken as 2*w whereas the right child will be taken as 2*w+1 because root node is present at index 0 in the array and to access every index position in the array.
5. 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
View Answer
Answer: a
Explanation: Floor of w-1/2 because we can't miss a node.

6. 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
View Answer
Answer: a
Explanation: Array cannot represent arbitrary shaped trees. It can only be used in case of complete trees. If every node stores data saying that which of its children exists in the array then elements can be accessed easily.
7. What must be the missing logic in place of missing lines for finding sum of nodes of binary tree in alternate levels?

```
  //e.g:-consider -complete binary tree:-height-3, [1,2,3,4,5,6,7]-answer
must be 23
  n=power(2,height)-1; //assume input is height and a[i] contains tree
elements
  for(i=1;i<=n;)
  {
        //present level is initialized to 1 and sum is initialized to  0
        for(j=1;j<=pow(2,currentlevel-1);j++)
        {
           sum=sum+a[i];
           i=i+1;
        }
     //missing logic
  }
```

a)

```
   i=i+pow(2,currentlevel);
   currentlevel=currentlevel+2;
   j=1;
```

b)

```
   i=i+pow(2,currentlevel);
   currentlevel=currentlevel+2;
   j=0;
```

c)

```
   i=i-pow(2,currentlevel);
   currentlevel=currentlevel+2;
   j=1;
```

d)

```
   i=i+pow(2,currentlevel);
   currentlevel=currentlevel+1;
   j=1;
```

View Answer
Answer: a
Explanation: The i value must skip through all nodes in the next level and current level must be one+next level.

8. 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 remaning cases it is fine
d) No linked list representation of tree is only fine
View Answer
Answer: c
Explanation: In case of sparse trees (where one node per level in worst cases), the array size $(2^h)$-1 where h is height but only h indexes will be filled and $(2^h)$-1-h nodes will be left unused leading to space wastage.
9. Why is heap implemented using array representations than tree(linked list) representations though both tree representations and heaps have same complexities?

```
for binary heap
-insert: O(log n)
-delete min: O(log n)

for a tree
-insert: O(log n)
-delete: O(log n)
```

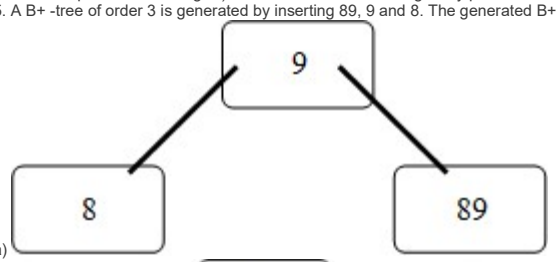
Then why go with array representation when both are having same values ?
a) arrays can store trees which are complete and heaps are not complete
b) lists representation takes more memory hence memory efficiency is less and go with arrays and arrays have better caching
c) lists have better caching

d) In lists insertion and deletion is difficult
View Answer
Answer: b
Explanation: In memory the pointer address for next node may not be adjacent or nearer to each other and also array have wonderful caching power from os and manipulating pointers is a overhead. Heap data structure is always a complete binary tree.
10. 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
View Answer
Answer: b
Explanation: We need any two traversals for tree formation but if some additional stuff or techniques are used while storing a tree in an array then one traversal can facilitate like also storing null values of a node in array.

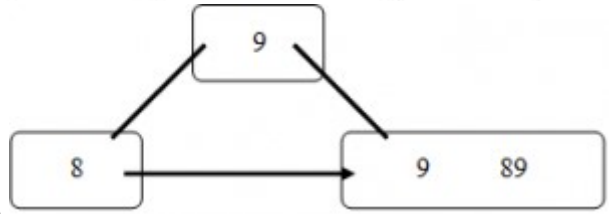
## B+ Tree Multiple Choice Questions and Answers (MCQs)

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “B+ Tree”.
1. In a B+ tree, both the internal nodes and the leaves have keys.
a) True
b) False
View Answer
Answer: b
Explanation: In a B+ -tree, only the leaves have keys, and these keys are replicated in non-leaf nodes for defining the path for locating individual records.
2. Which of the following is true?
a) B + tree allows only the rapid random access
b) B + tree allows only the rapid sequential access
c) B + tree allows rapid random access as well as rapid sequential access
d) B + tree allows rapid random access and slower sequential access
View Answer
Answer: c
Explanation: The B+ -tree being a variation of B-tree allows rapid random access. In a B+ -tree the leaves are linked together, so it also provides rapid sequential access.
3. A B+ tree can contain a maximum of 7 pointers in a node. What is the minimum number of keys in leaves?
a) 6
b) 3
c) 4
d) 7
View Answer
Answer: b
Explanation: Maximum number of pointers in a node is 7, i.e. the order of the B+ -tree is 7. In a B+ tree of order n each leaf node contains at most n – 1 key and at least ⌈(n − 1)/2⌉ keys. Therefore, a minimum number of keys each leaf can have = ⌈(7 – 1)/2⌉ = 3.
4. Which of the following is false?
a) A B+ -tree grows downwards
b) A B+ -tree is balanced
c) In a B+ -tree, the sibling pointers allow sequential searching
d) B+ -tree is shallower than B-tree
View Answer
Answer: a
Explanation: A B+ -tree always grows upwards. And In a B+tree – i)The path from the root to every leaf node is of the same length, so the tree is balanced. ii) Leaves are linked, so allow sequential searching. iii) An index is built with a single key per block of data rather than with one key per data record, so it is shallower than B-tree.
5. A B+ -tree of order 3 is generated by inserting 89, 9 and 8. The generated B+ -tree is _____

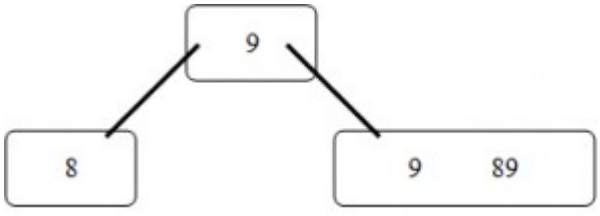
a)

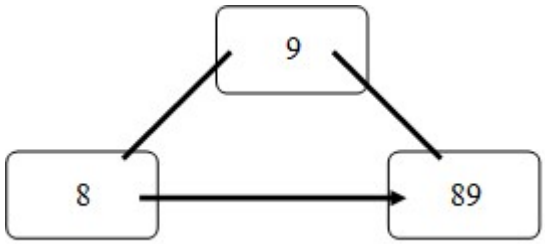



b)



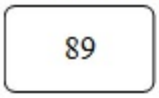

c)





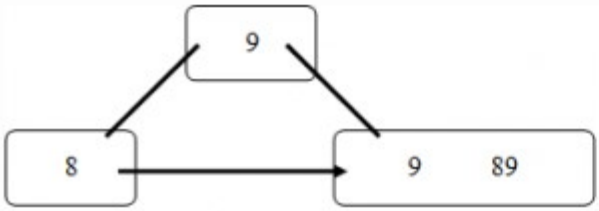

d)
View Answer
Answer: b
Explanation:

In a B+ -tree of order 3 every non-root node has at most 2 keys.
After inserting 89 the tree formed will be,




After inserting 9


After inserting 8, the node will get overflowed so we split the node and create a new root with key 9. So, the tree formed will be





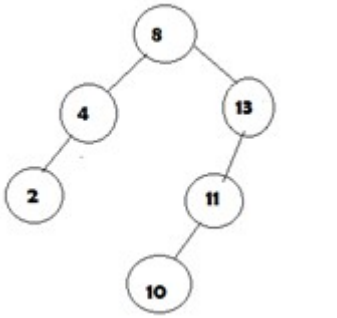

6. Statement 1: When a node is split during insertion, the middle key is promoted to the parent as well as retained in right half-node.
Statement 2: When a key is deleted from the leaf, it is also deleted from the non-leaf nodes of the tree.
a) Statement 1 is true but statement 2 is false
b) Statement 2 is true but statement 1 is false
c) Both the statements are true
d) Both the statements are false
View Answer
Answer: a
Explanation: During the split, the middle key is retained in the right half node and also promoted to parent node. When a key is deleted from the leaf, it is retained in non-leaves, because it can be still a valid separator between keys in nodes below.
7. Efficiency of finding the next record in B+ tree is ____
a) O(n)
b) O(log n)
c) O(nlog n)
d) O(1)
View Answer
Answer: d
Explanation: In a B+ -tree finding the next recored (successor) involves accessing an additional leaf at most. So, the efficiency of finding the next record is O(1).
8. What is the maximum number of keys that a B+ -tree of order 3 and of height 3 have?
a) 3
b) 80
c) 27
d) 26
View Answer
Answer: d
Explanation: A B+ tree of order n and height h can have at most $n^h$ – 1 keys. Therefore maximum number of keys = $3^3$ -1 = 27 -1 = 26.
9. Which of the following is false?
a) Compared to B-tree, B+ -tree has larger fanout
b) Deletion in B-tree is more complicated than in B+ -tree
c) B+ -tree has greater depth than corresponding B-tree
d) Both B-tree and B+ -tree have same search and insertion efficiencies
View Answer
Answer: c
Explanation: A B+ -tree has larger fanout and therefore have a depth smaller than that of corresponding B-tree.

10. Which one of the following data structures are preferred in database-system implementation?
a) AVL tree
b) B-tree
c) B+ -tree
d) Splay tree
View Answer
Answer: c
Explanation: The database-system implementations use B+ -tree data structure because they can be used for multilevel indexing.

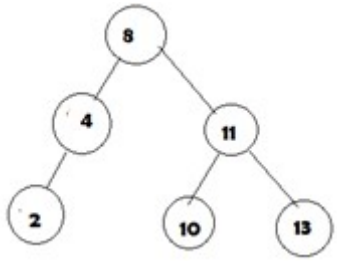
## Data Structure Questions and Answers – AVL Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “AVL Tree”.
1. What is an AVL tree?
a) a tree which is balanced and is a height balanced tree
b) a tree which is unbalanced and is a height balanced tree
c) a tree with three children
d) a tree with atmost 3 children
View Answer
Answer: a
Explanation: It is a self balancing tree with height difference atmost 1.
2. Why we need to a binary tree which is height balanced?
a) to avoid formation of skew trees
b) to save memory
c) to attain faster memory access
d) to simplify storing
View Answer
Answer: a
Explanation: In real world dealing with random values is often not possible, the probability that u are dealing with non random values(like sequential) leads to mostly skew trees, which leads to worst case. hence we make height balance by rotations.
3. Which of the below diagram is following AVL tree property?

i.




ii.




a) only i
b) only i and ii
c) only ii
d) i is not a binary search tree
View Answer
Answer: b
Explanation: The property of AVL tree is it is height balanced tree with difference of atmost 1 between left and right subtrees. All AVL trees are binary search tree.
4. What is the maximum height of an AVL tree with p nodes?
a) p
b) log(p)
c) log(p)/2
d) ½
View Answer
Answer: b
Explanation: Consider height of tree to be ‘he’, then number of nodes which totals to p can be written in terms of height as N(he)=N(he-1)+1+N(he-2). since N(he) which is p can be written in terms of height as the beside recurrence relation which on solving gives N(he)= O(logp) as worst case height.
5. To restore the AVL property after inserting a element, we start at the insertion point and move towards root of that tree. is this statement true?
a) true
b) false
View Answer
Answer: a
Explanation: It is interesting to note that after insertion, only the path from that point to node or only that subtrees are imbalanced interms of height.

6. Given an empty AVL tree, how would you construct AVL tree when a set of numbers are given without performing any rotations?
a) just build the tree with the given input
b) find the median of the set of elements given, make it as root and construct the tree
c) use trial and error
d) use dynamic programming to build the tree
View Answer
Answer: b
Explanation: Sort the given input, find the median element among them, make it as root and construct left and right subtrees with elements lesser and greater than the median element recursively. this ensures the subtrees differ only by height 1.

7. What maximum difference in heights between the leafs of a AVL tree is possible?
a) log(n) where n is the number of nodes
b) n where n is the number of nodes
c) 0 or 1
d) atmost 1
View Answer
Answer: a
Explanation: At every level we can form a tree with difference in height between subtrees to be atmost 1 and so there can be log(n) such levels since height of AVL tree is log(n).
8. Consider the pseudo code:

```
int avl(binarysearchtree root):
    if(not root)
      return 0
    left_tree_height = avl(left_of_root)

    if(left_tree_height== -1)
      return left_tree_height

    right_tree_height= avl(right_of_root)

    if(right_tree_height==-1)
      return right_tree_height
```

Does the above code can check if a binary search tree is an AVL tree?
a) yes
b) no
View Answer
Answer: a
Explanation: The condition to check the height difference between left and right subtrees is missing. if (absolute(left_tree_height – right_tree_height)>1) must be added.
9. Consider the below left-left rotation pseudo code where the node contains value pointers to left, right child nodes and a height value and Height() function returns height value stored at a particular node.

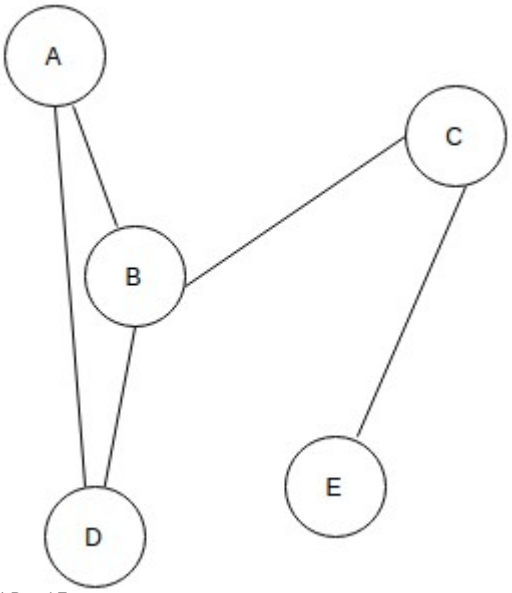
```
avltree leftrotation(avltreenode z):
  avltreenode w =x-left
  x-left=w-right
  w-right=x
  x-height=max(Height(x-left),Height(x-right))+1
  w-height=max(missing)+1
return w
```

What is missing?
a) Height(w-left), x-height
b) Height(w-right), x-height
c) Height(w-left), x
d) Height(w-left)
View Answer
Answer: a
Explanation: In the code we are trying to make the left rotation and so we need to find maximum of those two values.
10. Why to prefer red-black trees over AVL trees?
a) Because red-black is more rigidly balanced
b) AVL tree store balance factor in every node which costs space
c) AVL tree fails at scale
d) Red black is more efficient
View Answer
Answer: b
Explanation: Every node in an AVL tree need to store the balance factor (-1, 0, 1) hence space costs to O(n), n being number of nodes. but in red-black we can use the sign of number (if numbers being stored are only positive) and hence save space for storing balancing information. there are even other reasons where redblack is mostly prefered.

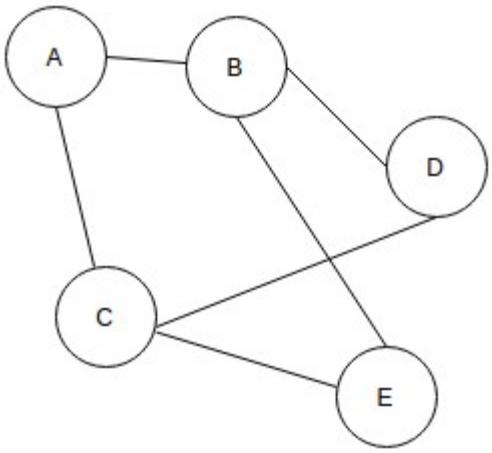
## Data Structure Questions and Answers – Graph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Graph”.
1. Which of the following statements for a simple graph is correct?
a) Every path is a trail
b) Every trail is a path
c) Every trail is a path as well as every path is a trail
d) Path and trail have no relation
View Answer
Answer: a
Explanation: In a walk if the vertices are distinct it is called a path, whereas if the edges are distinct it is called a trail.

2. In the given graph identify the cut vertices.

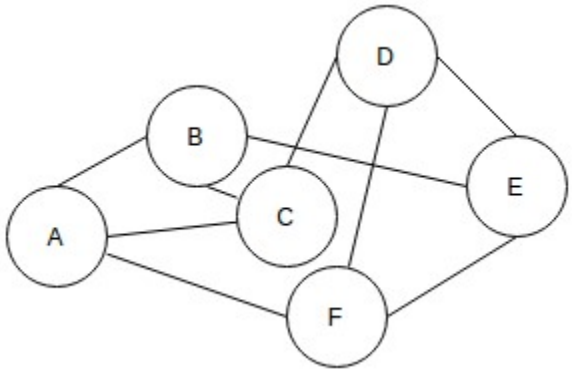



a) B and E
b) C and D
c) A and E
d) C and B
View Answer
Answer: d
Explanation: After removing either B or C, the graph becomes disconnected.
3. For the given graph(G), which of the following statements is true?




a) G is a complete graph
b) G is not a connected graph
c) The vertex connectivity of the graph is 2
d) The edge connectivity of the graph is 1
View Answer
Answer: c
Explanation: After removing vertices B and C, the graph becomes disconnected.
4. What is the number of edges present in a complete graph having n vertices?
a) (n*(n+1))/2
b) (n*(n-1))/2
c) n
d) Information given is insufficient
View Answer
Answer: b
Explanation: Number of ways in which every vertex can be connected to each other is nC2.

5. The given Graph is regular.




a) True
b) False
View Answer
Answer: a
Explanation: In a regular graph, degrees of all the vertices are equal. In the given graph the degree of every vertex is 3.

6. In a simple graph, the number of edges is equal to twice the sum of the degrees of the vertices.
a) True
b) False
View Answer
Answer: b
Explanation: The sum of the degrees of the vertices is equal to twice the number of edges.
7. A connected planar graph having 6 vertices, 7 edges contains _____ regions.
a) 15
b) 3
c) 1
d) 11
View Answer
Answer: b
Explanation: By euler's formula the relation between vertices(n), edges(q) and regions(r) is given by n-q+r=2.
8. If a simple graph G, contains n vertices and m edges, the number of edges in the Graph G'(Complement of G) is _____
a) (n*n-n-2*m)/2
b) (n*n+n+2*m)/2
c) (n*n-n-2*m)/2
d) (n*n-n+2*m)/2
View Answer
Answer: a
Explanation: The union of G and G' would be a complete graph so, the number of edges in G'= number of edges in the complete form of G(nC2)-edges in G(m).
9. Which of the following properties does a simple graph not hold?
a) Must be connected
b) Must be unweighted
c) Must have no loops or multiple edges
d) Must have no multiple edges
View Answer
Answer: a
Explanation: A simple graph maybe connected or disconnected.
10. What is the maximum number of edges in a bipartite graph having 10 vertices?
a) 24
b) 21
c) 25
d) 16
View Answer
Answer: c
Explanation: Let one set have n vertices another set would contain 10-n vertices.
Total number of edges would be n*(10-n), differentiating with respect to n, would yield the answer.
11. Which of the following is true?
a) A graph may contain no edges and many vertices
b) A graph may contain many edges and no vertices
c) A graph may contain no edges and no vertices
d) A graph may contain no vertices and many edges
View Answer
Answer: b
Explanation: A graph must contain at least one vertex.
12. For a given graph G having v vertices and e edges which is connected and has no cycles, which of the following statements is true?
a) v=e
b) v = e+1
c) v + 1 = e
d) v = e-1
View Answer
Answer: b
Explanation: For any connected graph with no cycles the equation holds true.
13. For which of the following combinations of the degrees of vertices would the connected graph be eulerian?
a) 1,2,3
b) 2,3,4
c) 2,4,5
d) 1,3,5
View Answer
Answer: a
Explanation: A graph is eulerian if either all of its vertices are even or if only two of its vertices are odd.
14. A graph with all vertices having equal degree is known as a _____
a) Multi Graph
b) Regular Graph

c) Simple Graph
d) Complete Graph
View Answer
Answer: b
Explanation: The given statement is the definition of regular graphs.
15. Which of the following ways can be used to represent a graph?
a) Adjacency List and Adjacency Matrix
b) Incidence Matrix
c) Adjacency List, Adjacency Matrix as well as Incidence Matrix
d) No way to represent
View Answer
Answer: c
Explanation: Adjacency Matrix, Adjacency List and Incidence Matrix are used to represent a graph.