[**Self Promotion**](https://github.com/panchalbhavya2210)

**Q - 1 : DEFINITION LINEAR DATA STRUCTURE IN DETAIL.**

ANS->

**Linear Data Structure**

It is a type of data structure where the arrangement of the data follows a linear trend. The data elements are arranged linearly such that the element is directly linked to its previous and the next elements. As the elements are stored linearly, the structure supports single-level storage of data. And hence, traversal of the data is achieved through a single run only.

Characteristics

* It is a type of data structure where data is stored and managed in a linear sequence.
* Data elements in the sequence are linked to one after the other.
* Implementation of the linear structure of data in a computer’s memory is easy as the data is organized sequentially.
* Array, queue. Stack, linked list, etc. are examples of this type of structure.
* The data elements stored in the data structure have only one relationship.
* Traversal of the data elements can be carried out in a single run as the data elements are stored in a single level.
* There is poor utilization of the computer memory if a structure storing data linearly is implemented.
* With the increase in the size of the data structure, the time complexity of the structure increases.

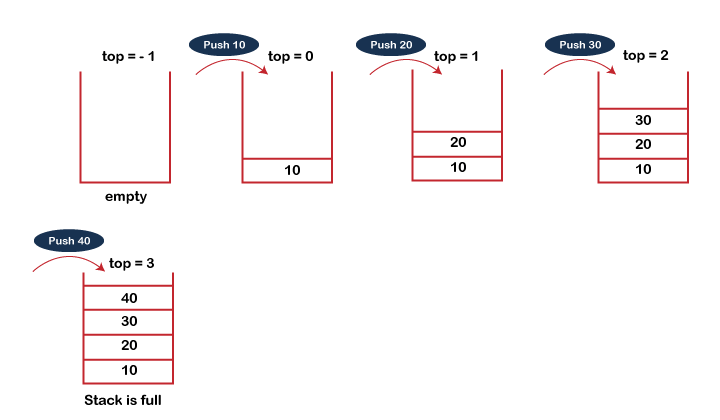
**Q - 2: EXPLAIN PUSH AND POP ALGORITHM IN DETAIL.**

ANS ->

**PUSH operation**

The steps involved in the PUSH operation is given below:

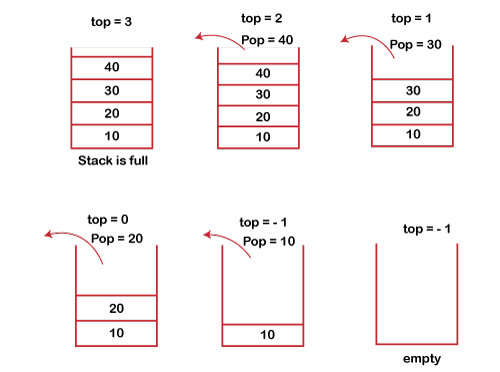
* Before inserting an element in a stack, we check whether the stack is full.
* If we try to insert the element in a stack, and the stack is full, then the overflow condition occurs.
* When we initialize a stack, we set the value of top as -1 to check that the stack is empty.
* When the new element is pushed in a stack, first, the value of the top gets incremented, i.e., top=top+1, and the element will be placed at the new position of the top.
* The elements will be inserted until we reach the max size of the stack.



**POP operation**

The steps involved in the POP operation is given below:

* Before deleting the element from the stack, we check whether the stack is empty.
* If we try to delete the element from the empty stack, then the underflow condition occurs.
* If the stack is not empty, we first access the element which is pointed by the top
* Once the pop operation is performed, the top is decremented by 1, i.e., top=top-1.



**Q - 3: CONVERT INFIX INTO PROFIX & PREFIX**

ANS ->

**PROFIX**

Given two operands a and b and an operator \odot , the infix notation implies that O will be placed in between a and b i.e a \odot b . When the operator is placed after both operands i.e ab\odot , it is called postfix notation. And when the operator is placed before the operands i.e \odot a b , the expression in prefix notation.

Given any infix expression, we can obtain the equivalent prefix and postfix format.

Input : A \* B + C / D

Output : + \* A B/ C D

Input : (A - B/C) \* (A/K-L)

Output : \*-A/BC-/AKL

**PREFIX**

To convert infix expression to postfix expression, we will use the stack data structure. By scanning the infix expression from left to right, when we will get any operand, simply add them to the postfix form, and for the operator and parenthesis, add them in the stack maintaining the precedence of them.

Input:

The infix expression. x^y/(5\*z)+2

Output:

Postfix Form Is: xy^5z\*/2+

**Q - 4: DEFINE QUEUE AND TYPES OF QUEUE IN DETAILS.**

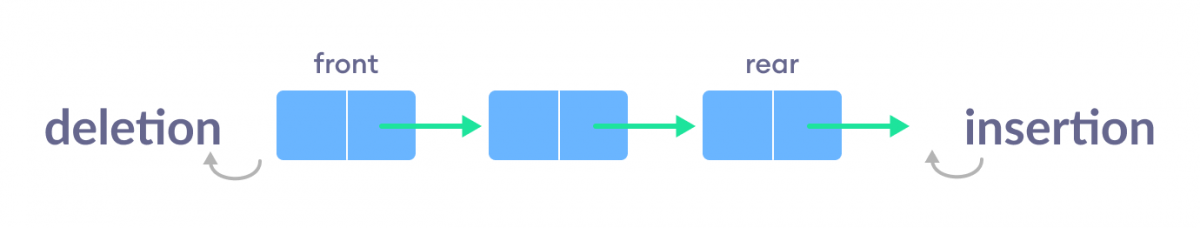
ANS -> A queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket.

There are four different types of queues:

* Simple Queue
* Circular Queue
* Priority Queue
* Double Ended Queue

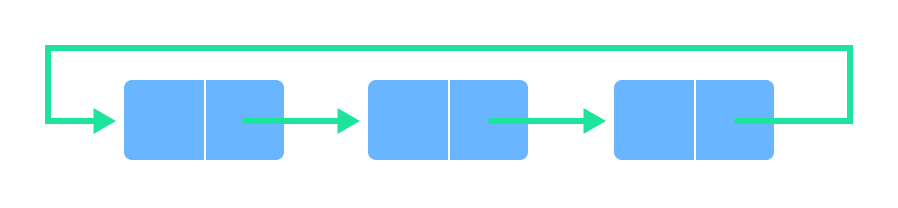
**Simple Queue**

In a simple queue, insertion takes place at the rear and removal occurs at the front. It strictly follows the FIFO (First in First out) rule.



**Circular Queue**

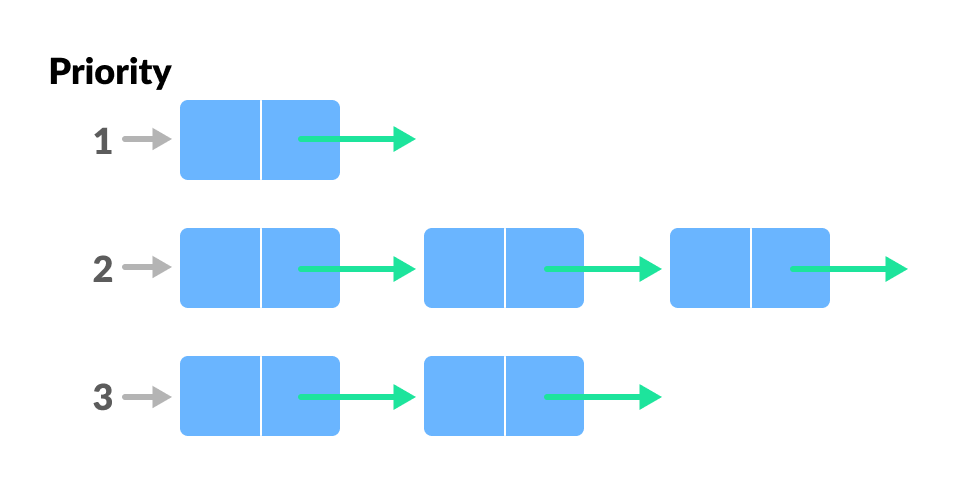
In a circular queue, the last element points to the first element making a circular link.



The main advantage of a circular queue over a simple queue is better memory utilization. If the last position is full and the first position is empty, we can insert an element in the first position. This action is not possible in a simple queue.

**Priority Queue**

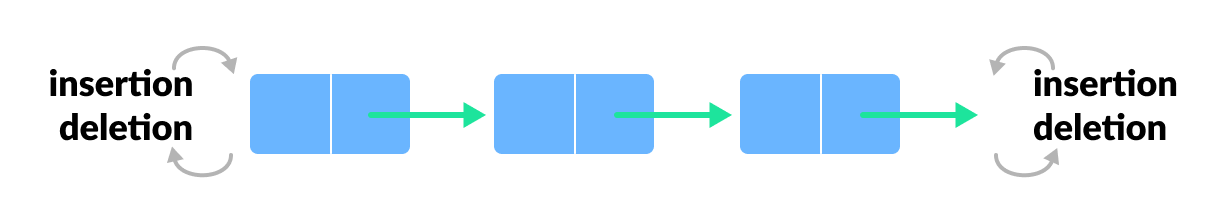
A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority. If elements with the same priority occur, they are served according to their order in the queue.



Insertion occurs based on the arrival of the values and removal occurs based on priority.

**Deque (Double Ended Queue)**

In a double ended queue, insertion and removal of elements can be performed from either from the front or rear. Thus, it does not follow the FIFO (First In First Out) rule.



**Q - 5: DEFINE LINKED LIST AND TYPES OF LINKED LIST.**

ANS -> A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

Types of Linked List

Following are the various types of linked list.

* **Simple Linked List** − Item navigation is forward only.
* **Doubly Linked List** − Items can be navigated forward and backward.
* **Circular Linked List** − Last item contains link of the first element as next and the first element has a link to the last element as previous.

**Q - 6 : ALGORITHM OF INSERTION AT FIRST IN SINGLY LINKED LINK.**

ANS ->

**Algorithm**

Step 1: IF PTR = NULL

Write OVERFLOW

Go to Step 7

[END OF IF]

Step 2: SET NEW\_NODE = PTR

Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW\_NODE → DATA = VAL

Step 5: SET NEW\_NODE → NEXT = HEAD

Step 6: SET HEAD = NEW\_NODE

Step 7: EXIT

**Q - 7 : ALGORITHM OF INSERTION AT FIRST IN DOUBLING LINKED LINK.**

ANS ->

**Algorithm** :

Step 1: IF ptr = NULL

Write OVERFLOW

Go to Step 9

[END OF IF]

Step 2: SET NEW\_NODE = ptr

Step 3: SET ptr = ptr -> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET NEW\_NODE -> PREV = NULL

Step 6: SET NEW\_NODE -> NEXT = START

Step 7: SET head -> PREV = NEW\_NODE

Step 8: SET head = NEW\_NODE

Step 9: EXIT

**Q - 8 : ALGORITHM OF INSERTION AT FIRST & LAST CIRCULAR LINKED LINK.**

ANS -> **Algorithm**

Step 1: IF PTR = NULL

Write OVERFLOW

Go to Step 11

[END OF IF]

Step 2: SET NEW\_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET TEMP = HEAD

Step 6: Repeat Step 8 while TEMP -> NEXT != HEAD

Step 7: SET TEMP = TEMP -> NEXT

[END OF LOOP]

Step 8: SET NEW\_NODE -> NEXT = HEAD

Step 9: SET TEMP → NEXT = NEW\_NODE

Step 10: SET HEAD = NEW\_NODE

Step 11: EXIT

**Q-9 EXPLAIN OPERATION WE CAN PERFORM ON QUEUE.**

ANS ->

**Different operations in Queue Data Structure:**

The various operations that are supported by a queue data structure that helps the user to modify and manipulate the data present in the queue are:

* **Enqueue operation**: The term "enqueue" refers to the act of adding a new element to a queue. Where does a new individual go and wait in a standard queue at a ticket counter to join the queue? The individual walks to the back of the room and takes a seat. A new element in a queue is similarly added at the end of the queue.
* **Dequeue operation**: Dequeue is the process of deleting an item from a queue. We must delete the queue member that was put first since the queue follows the FIFO principle. We'll delete the front element and make the element behind it the new front element because the element added initially will naturally be at the head of the queue.
* **Front Operation:** This works similarly to the peek operation in stacks in that it returns the value of the first element without deleting it.
* **isEmpty Operation**: The isEmpty() function is used to check if the Queue is empty or not.
* **isFull Operation**: The isFull() function is used to check if the Queue is full or not.