A linked list is a sequence of data structures, which are connected together via links. Linked list is the second most-used data structure after array. Linked Lists are used to create trees and graphs.

## Linked List Representation



As per the above illustration, following are the important points to be considered.

* Linked List contains a link element called first.
* Each link carries a data element and a link field called next.
* Each link is linked with its next link using its next link.
* Last link carries a link as null to mark the end of the list.

## Basic Operations

1. Insertion at beginning: It involves inserting any element at the front of the list. We just need a few link adjustments to make the new node as the head of the list.
2. Insertion at end of the list: It involves insertion at the last of the linked list. The new node can be inserted as the only node in the list or it can be inserted as the last one. Different logics are implemented in each scenario.
3. Insertion after specified node: It involves insertion after the specified node of the linked list. We need to skip the desired number of nodes in order to reach the node after which the new node will be inserted.
4. Deletion at beginning: It involves deletion of a node from the beginning of the list. This is the simplest operation among all. It just needs a few adjustments in the node pointers.
5. Deletion at the end of the list: It involves deleting the last node of the list. The list can either be empty or full. Different logic is implemented for the different scenarios.
6. Deletion after specified node: It involves deleting the node after the specified node in the list. we need to skip the desired number of nodes to reach the node after which the node will be deleted. This requires traversing through the list.
7. Traversing: In traversing, we simply visit each node of the list at least once in order to perform some specific operation on it, for example, printing data part of each node present in the list.
8. Searching: In searching, we match each element of the list with the given element. If the element is found on any of the locations then location of that element is returned otherwise null is returned.

# Linear Linked List

One way chain or singly linked list can be traversed only in one direction. In other words, we can say that each node contains only the next pointer, therefore we can not traverse the list in the reverse direction.

# Doubly Linked List

In a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer) , pointer to the previous node (previous pointer). Items can be navigated forward and backward.

# Circular Linked List

In a circular Singly linked list, the last node of the list contains a pointer to the first node of the list. We traverse a circular singly linked list until we reach the same node where we started. The circular singly linked list has no beginning and no ending. There is no null value present in the next part of any of the nodes.

## Doubly Linked List as Circular

In the doubly linked list, the next pointer of the last node points to the first node and the previous pointer of the first node points to the last node making the circular in both directions.