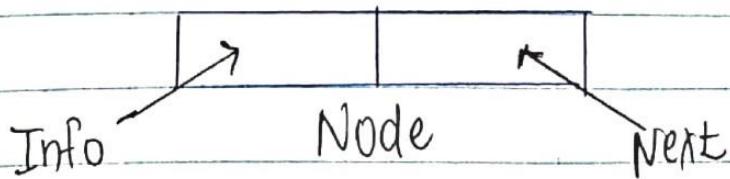
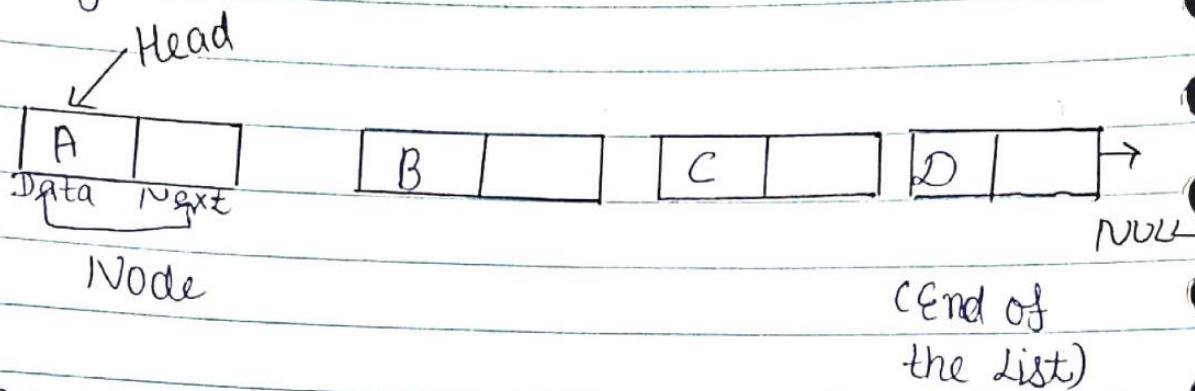


Linked Lists

- A linked list is a linear data structure. It can be defined as the linear collection of elements where each element stored in a node.
 - In this, data are not stored sequentially inside the memory but they are link with each other by the help of address.
 - Elements are not stored at contiguous Memory location.
 - Node has two parts:-
 - Info (Data)
 - Next (Pointer)
- Structure of node:-
- 
- The diagram shows a horizontal rectangular box divided into three equal-width sections. The left section is labeled 'Info' with an arrow pointing to its center. The middle section is labeled 'Node' with an arrow pointing to its center. The right section is labeled 'Next' with an arrow pointing to its center. There are small arrows at the top and bottom corners of the box, pointing towards the center of each section.

Each node contains two fields : i.e. data stored at particular address and the which contains

the address of the **next** node in the memory.



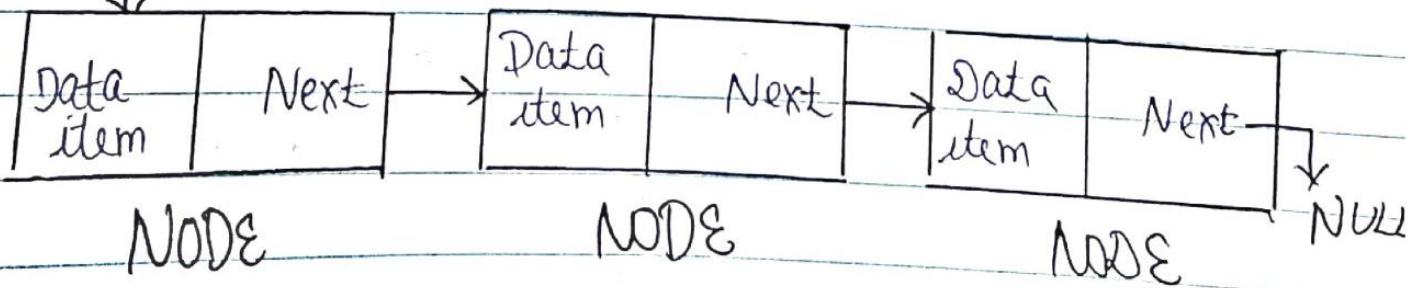
- In simple words, a linked list consists of nodes where each node contains a **data** field and a reference link to the next node in the list.



Representation of Linked List :-

Linked List can be represented as a chain of nodes, where every node points to next node.

Head ↴



- Linked List contains a start pointer, holds the address of the first node of the list. It is also called **head** Pointer.
- Each node carries a data field and a link field called next.
- Each node is linked with its next link using its next link.
- Null Pointer indicates end of the list i.e. last node of the list.
- The arrows are used to refer to the next node in the link list.

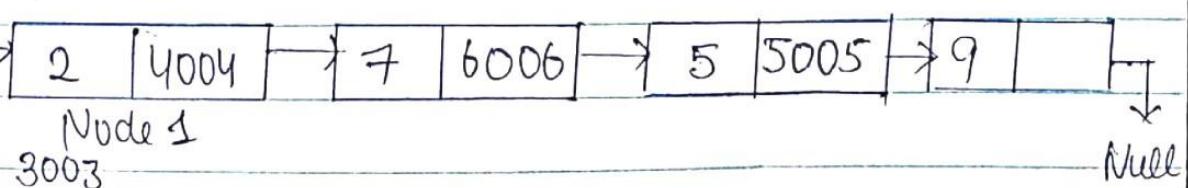
Example

Node	Data	Address
Node 1	2	3003
Node 2	7	4004
Node 3	5	6006
Node 4	9	5005

Link list will look like:-

START

3003



* Representation of Linked List in Memory:-

- Representation of linear linked list in memory requires two linear array:-
INFO → which store data elements
LINK → which store next pointer field.
- It also requires a variable START or HEAD which contains the address of first element.

Example:-

INFO [9] = INDIA	LINK [9] = 6
INFO [5] = UK	LINK [5] = 2
INFO [6] = USA	LINK [6] = 4
INFO [3] = AUS	LINK [3] = 7
INFO [4] = Newzealand	LINK [4] = 5
INFO [2] = CHINA	LINK [2] = 3
INFO [1] = JAPAN	LINK [7] = NULL

Solution:-

INDIA,

Representation of linked list in memory is :-

S.NO	INFO	LINK
1	-	-
2	CHINA	3
3	AUS	7
4	New Zealand	5
5	UK	2
6	USA	4
7	JAPAN	NULL
8	INDIA	-
START → 9	INDIA	6

* Operations of linked List :-

There are various operations that can be performed on link list :-

1. Creation
2. Traversing
3. Searching
4. Insertion
5. Deletion.
6. Sorting.

1. Creation :- Creating the linked list.

2. Traversal:- access each element of the linked list.

3. Insertion:- adds a new element to the linked list

4. Deletion:- removes the existing elements.

5. Search:- find a node in the linked list.

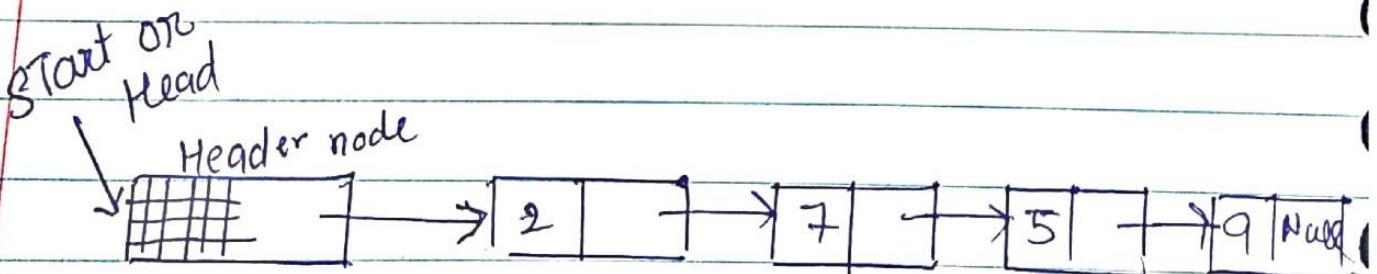
6. Sort:- sort the nodes of the linked list.

* Types of Linked List

- (i) Header linked List (Single)
- (ii) Doubly linked list
- (iii) Circular linked list

(i) Header linked list :-

- It is a linked list with a special node in the beginning of the list.
- This special node is known as header node.
- The pointer variable START/ HEAD points to the header node in header link list.
- Next field of header pointer to the first node in the link list.



(Header link with four nodes
having data 2,7,5 and 9)

Start → next = null

Header Node

Start → [] [] [] null

② Doubly linked list :-

- In doubly linked list or two-way list we can move forward as well as backward from a particular node.
- In doubly linked list each node is divided into three parts :-

(I) The first part (Address of Previous node)

(II) The second part (Data or Record)

(III) The third part (Address of next node)

node = Prev | Data | next



first

node

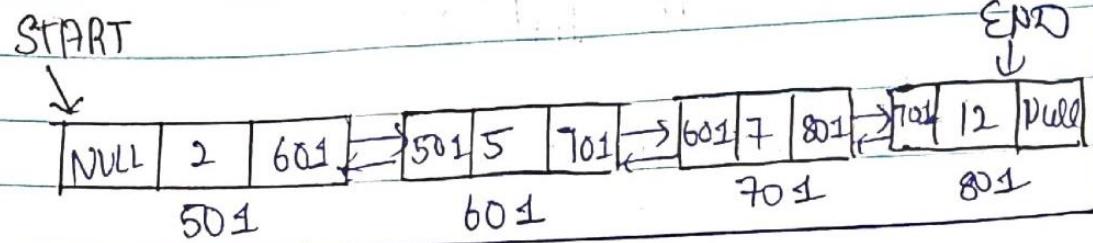
will be
null

Last

node

will be
null

Example:-



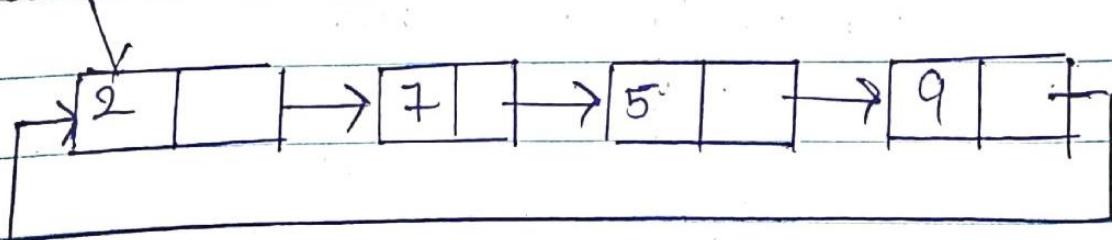
②

Circular linked list :-

- o Circular linked list is a list in which last node points to the first node of the list instead of Pointing NULL.
- o It means that Pointer Part of last node contains the address of first node.

Example:-

Start or Head



(Circular linked list)