Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
2	3

CS8391

DATA STRUCTURES
(Common to CSE & IT)

UNIT No. 1

LINEAR DATA STRUCTURES

1.1 LINKED LIST IMPLEMENTATION -SINGLE LINKED LIST

Version: 1.XX

Singly Linked List

Singly Linked List: A linked list allocates space for each element separately in its own block of memory called a "node". The list gets an overall structure by using pointers to connect all its nodes together like the links in a chain. Each node contains two fields; a "data" field to store whatever element, and a "next" field which is a pointer used to link to the next node. Each node is allocated in the heap using malloc(), so the node memory continues to exist until it is explicitly de- allocated using free(). The front of the list is a pointer to the “start” node

Why Linked List?

Even though **searching** an array for an **individual** element can be **very efficient**, array has some limitations. So arrays are generally not used to implement Lists**.**

Advantages of Linked List

1. Linked lists are dynamic data structures - The size is not fixed. They can grow or shrink during the execution of a program.

2. Efficient memory utilization - memory is not pre-allocated. Memory is allocated, whenever it is required and it is de- allocated whenever it is not needed. Data is stored in non-contiguous memory blocks.

3. Insertion and deletion of elements are easier and efficient.

Provides flexibility. No need to shift elements of a linked list to make room for a new element or to delete an element.

Disadvantages of Linked List

1. More memory - Needs space for pointer (link field).

2. Accessing arbitrary element is time consuming. Only sequential search is supported not binary search.

Operations on Linked List

The primitive operations performed on the linked list are as follows

1. **Creation**- This operation is used to create a linked list. Once a linked list is created with one node, insertion operation can be used to add more elements in a node.

2. **Insertion**- This operation is used to insert a new node at any specified location in the linked list. A new node may be inserted,

    1. At the beginning of the linked list,
    2. At the end of the linked list,
    3. At any specified position in between in a linked list.

3. **Deletion**- This operation is used to delete an item (or node) from the linked list. A node may be deleted from the,

    1. Beginning of a linked list,
    2. End of a linked list,
    3. Specified location of the linked list.

4. **Traversing** - It is the process of going through all the nodes from one end to another end of a linked list. In a singly linked list we can visit the nodes only from left to right (forward traversing). But in doubly linked list forward and backward traversing is possible.

5. **Searching**- It is the process finding a specified node in a linked list.

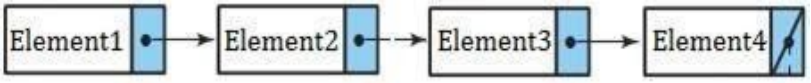
6. **Concatenation**- It is the process of appending the second list to the end of the first list. Consider a list A having n nodes and B with m nodes. Then the

operation concatenation will place the 1st node of B in the (n+1) the node in A. After concatenation A will contain (name) nodes.

Types of linked list

1. Singly Linked list or Linear list or One-way list
2. Doubly linked list or Two-way list
3. Circular linked list
4. Doubly circular linked list

In singly linked list, each element (except the first one) has a unique predecessor, and each element (except the last one) has a unique successor. Each node contains two parts: **data** or **Info** and **link**. The data holds the application data to be processed. The link contains the address of the next node in the list. That is, each node has a single pointer to the next node. The last node contains a NULL pointer indicating the end of the list.



SentinelNode

It is also called as **Header node**or **Dummy node.**

Advantages

Sentinel node is used to solve the following problems

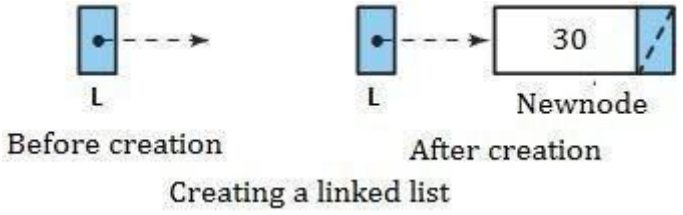
- First, there is **no really obvious way to insert at the front** of the list from the definitions given.
- Second, **deleting from the front of the list** is a special case, because it changes the start of the list; careless coding will lose the
- list.
- A third problem concerns deletion in general. Although the pointer moves above are simple, the deletion algorithm requires us to **keep track of the cell before the one that we want to delete.**

Disadvantages

o It consumes extra space.

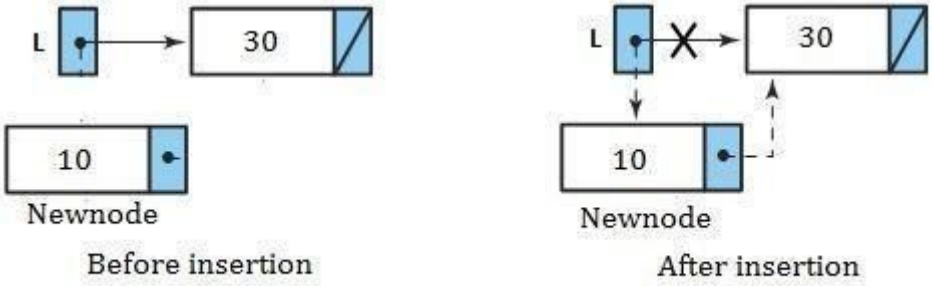
Insertion

a. Creating a new node from empty List



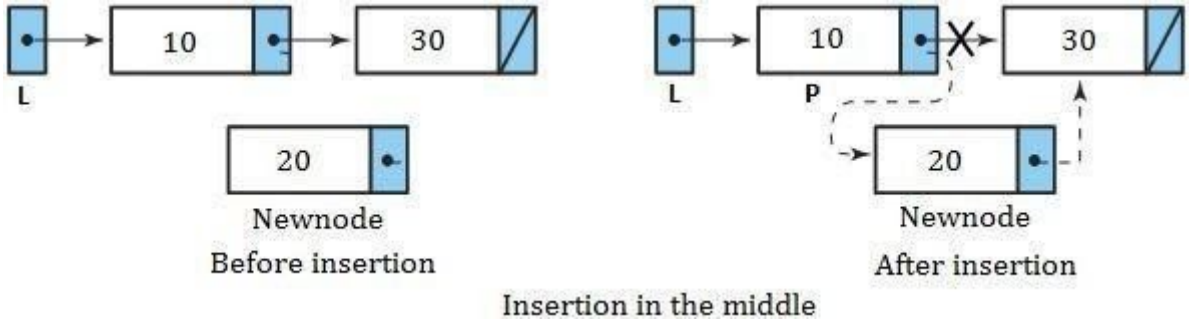
Creating a linked list

b.Inserting a node to the front of list



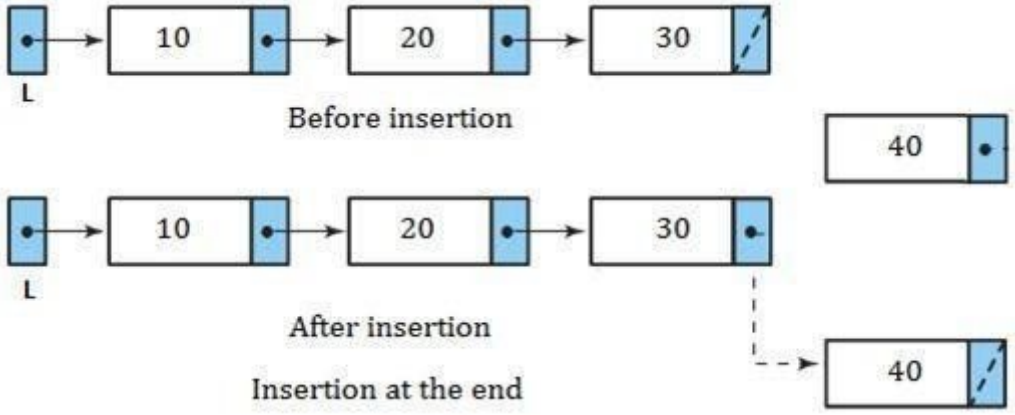
Insertion at the beginning

c. Inserting a node in the middle



Insertion in the middle

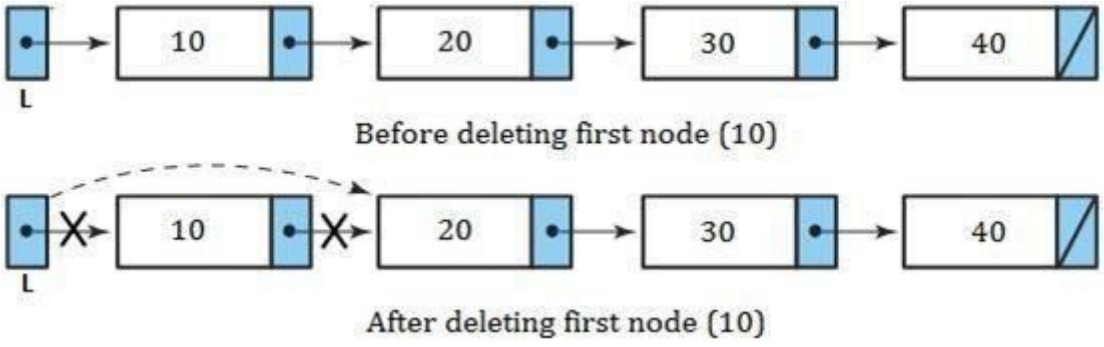
D. Inserting a node to the end of list



Insertion at the end

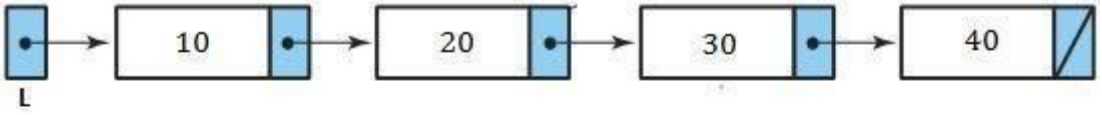
Deletion

a.Deleting a node to the front of list

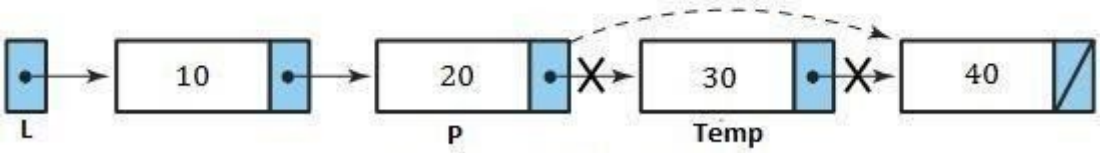


After deleting first node (10)

a. Deleting the middle node

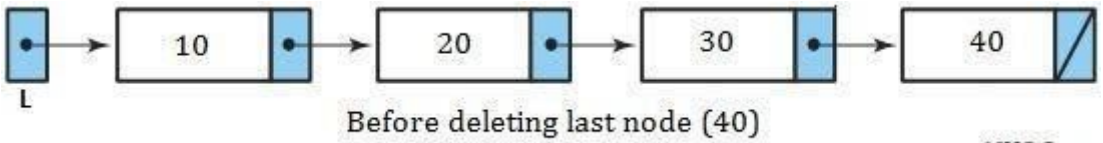


Before deleting middle node (30)

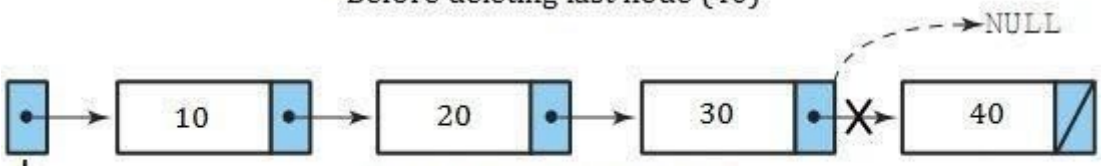


After deleting middle node (30)

b. Deleting the last node



Before deleting last node (40)



After deleting last node (40)