LINKED LISTS

A linked list is a linear collection of specially designed data elements, called nodes, linked to one another by means of pointers. Each node is divided into two parts: the first part contains the information of the element, and the second part contains the address of the next node in the linked list. Address part of the node is also called linked or next field. Following Fig shows a typical example of node.

Fig. shows a schematic diagram of a linked list with 3 nodes. Each node is pictured with two parts. The left part of each node contains the data items and the right part represents the address of the next node; there is an arrow drawn from it to the next node. The next pointer of the last node contains a special value, called the NULL pointer, which does not point to any address of the node. That is NULL pointer indicates the end of the linked list. START pointer will hold the address of the 1st node in the list START = NULL if there is no list (i.e.; NULL list or empty list).

TYPES OF LINKED LIST

Basically we can divide the linked list into the following three types in the order in which they (or node) are arranged:
1. Singly linked list
2. Doubly linked list
3. Circular linked list
1. SINGLY LINKED LIST
All the nodes in a singly linked list are arranged sequentially by linking with a pointer. A singly linked list can grow or shrink, because it is a dynamic data structure. Following figure explains the different operations on a singly linked list.

![Singly Linked List Diagram]

2. DOUBLY LINKED LIST
A doubly linked list is one in which all nodes are linked together by multiple links which help in accessing both the successor (next) and predecessor (previous) node for any arbitrary node within the list. Every node in the doubly linked list has three fields: LeftPointer, RightPointer and DATA. Fig. below shows a typical doubly linked list:

![Doubly Linked List Diagram]
LPoint will point to the node in the left side (or previous node) that is LPoint will hold the address of the previous node. RPoint will point to the node in the right side (or next node) that is RPoint will hold the address of the next node. DATA will store the information of the node.

3. CIRCULAR LINKED LIST

A circular linked list is one, which has no beginning and no end. A singly linked list can be made a circular linked list by simply storing the address of the very first node in the linked field of the last node. A circular linked list is shown in Fig. below:
A circular doubly linked list has both the successor pointer and predecessor pointer in circular manner as shown in the Fig. below:

![Circular Linked list](image)

![Circular Doubly Linked list](image)

**ADVANTAGES AND DISADVANTAGES OF LISTS:**

**Linked list have many advantages and some of them are:**

1. Linked list are dynamic data structure. That is, they can grow or shrink during the execution of a program.
2. Efficient memory utilization: In linked list (or dynamic) representation, memory is not pre-allocated. Memory is allocated whenever it is required. And it is deallocated (or removed) when it is not needed.
3. Insertion and deletion are easier and efficient. Linked list provides flexibility in inserting a data item at a specified position and deletion of a data item from the given position.
4. Many complex applications can be easily carried out with linked list.

**Linked list has following disadvantages**

1. More memory: to store an integer number, a node with integer data and address field is allocated. That is more memory space is needed.
2. Access to an arbitrary data item is little bit cumbersome and also time consuming.