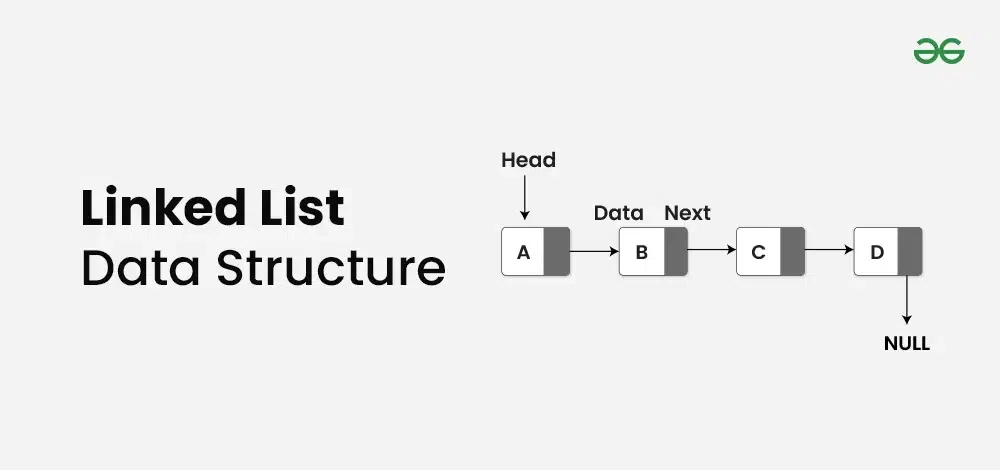
It consists of nodes where each node contains **data** and a**reference (link)** to the next node in the sequence.

This allows for dynamic memory allocation and efficient **insertion** and **deletion** operations compared to arrays.



**What is a Linked List?**

A **linked list**is a linear data structure that consists of a series of nodes connected by pointers.

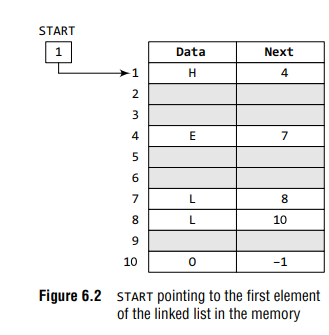
Each node contains **data** and a **reference(link)** to the next node in the list. Unlike **arrays, linked lists**allow for efficient **insertion** or **removal** of elements from any position in the list, as the nodes are not stored contiguously in memory.

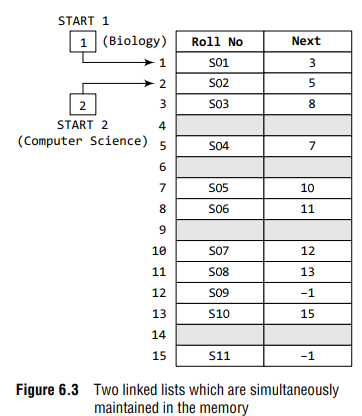
**Linked List:**

* **Data Structure:** Non-contiguous
* **Memory Allocation:** Dynamic
* **Insertion/Deletion:** Efficient
* **Access:** Sequential

**Array:**

* **Data Structure:** Contiguous
* **Memory Allocation:** Static
* **Insertion/Deletion:** Inefficient
* **Access:** Random



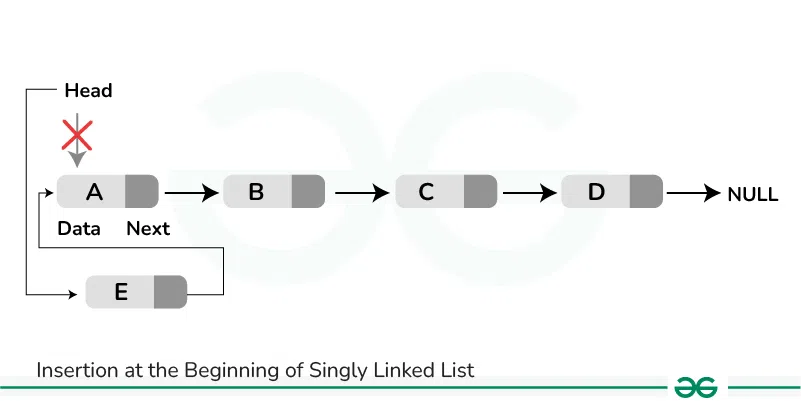


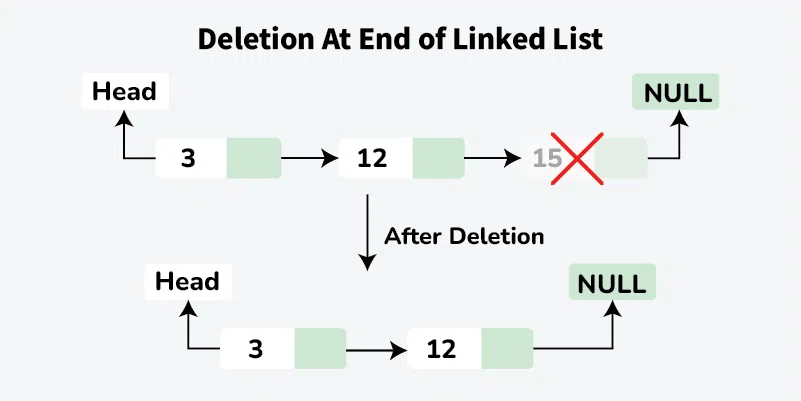
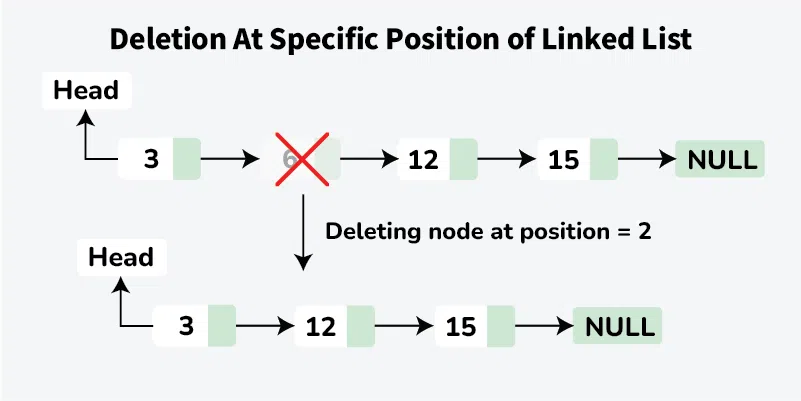
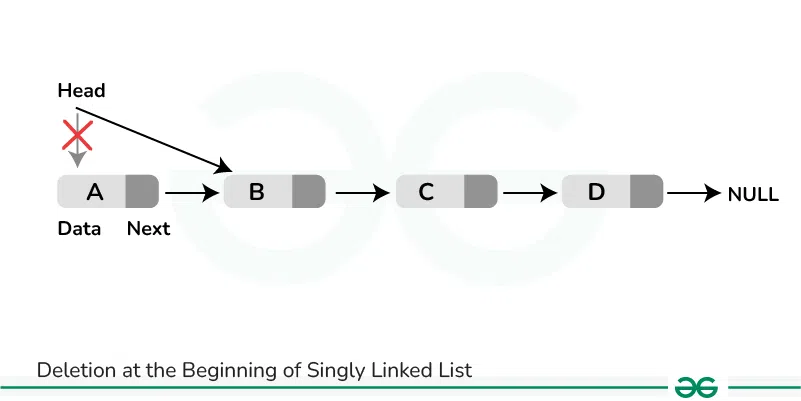
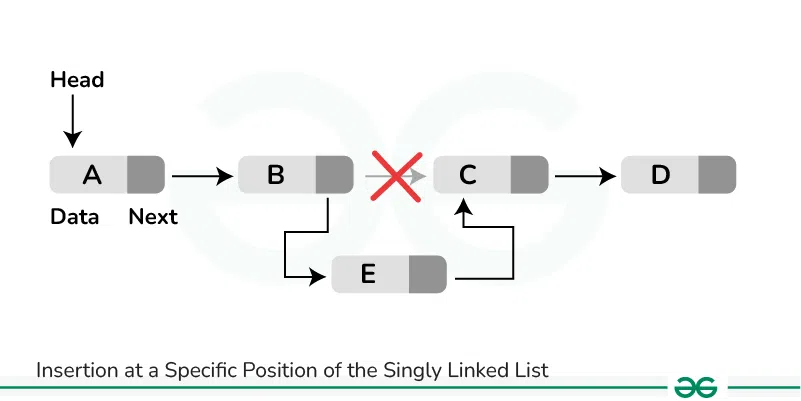
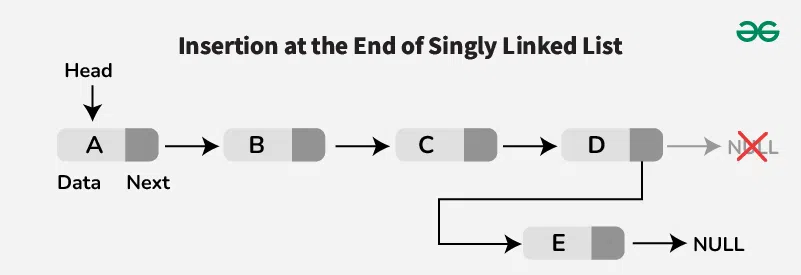
**What is Singly Linked List?**

*A****singly linked list****is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element using a pointer.*

***In singly linked list Link(pointer linking) hamesa aage ki taraf hota hai peeche ki taraf nhi***

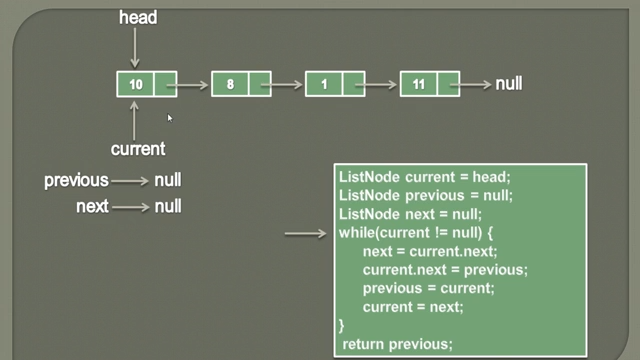
## [**How to Insert a Node at the Front/Beginning of Linked List**](https://www.geeksforgeeks.org/insert-a-node-at-front-beginning-of-a-linked-list/)



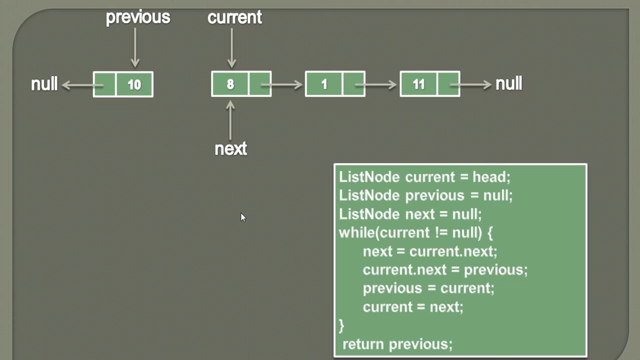
****

**Reverse a linked list**



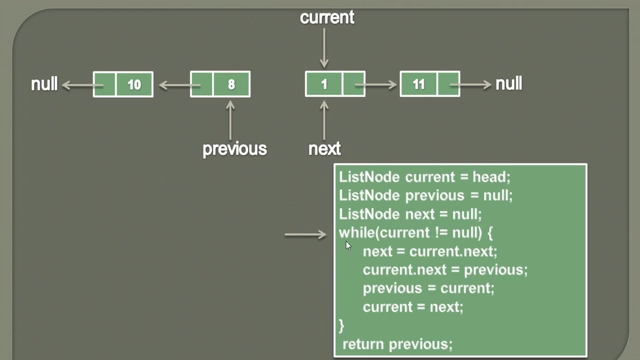
****

After 1st iteration

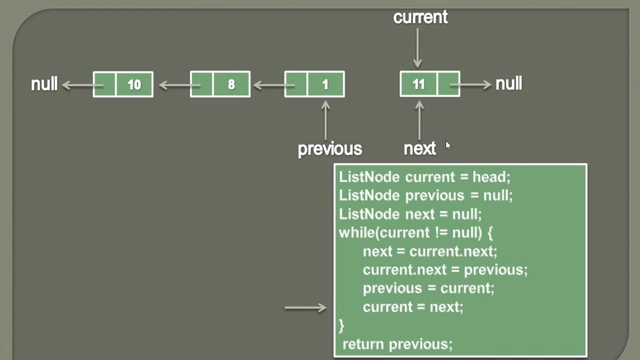




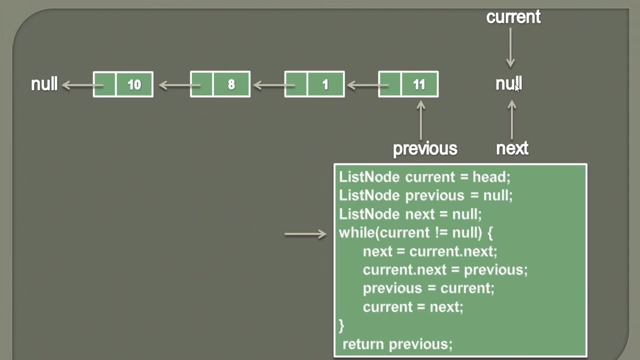
After 2nd iteration





After 3rd iteration

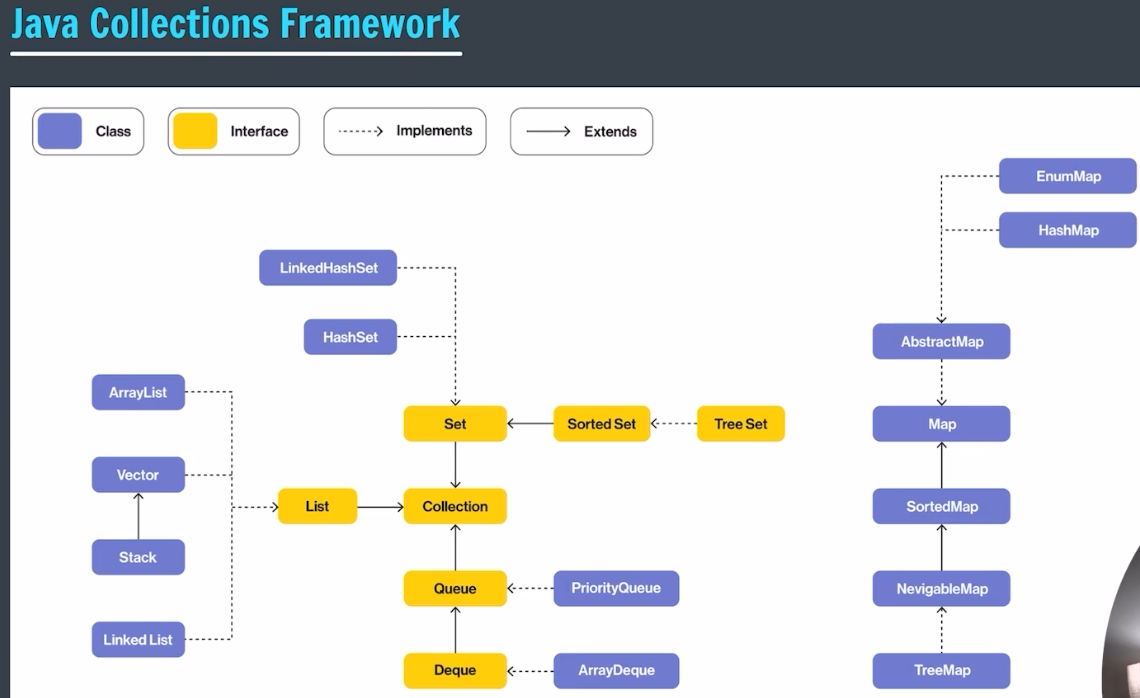


After 4th iteration

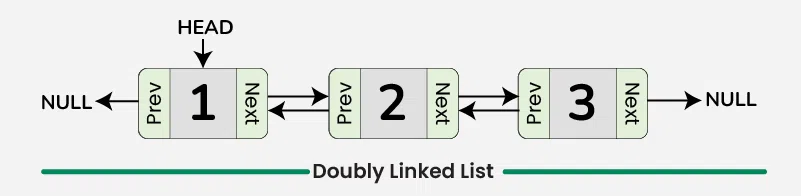


**Time Complexity: O(N)**The code **traverses** the **entire** **linked** **list** once, where 'n' is the number of nodes in the list. This traversal has a **linear** **time** **complexity**, O(n).

**Space Complexity: O(1)** The code uses only a **constant** **amount** of **additional** **space**, regardless of the linked list's length. This is achieved by using three pointers (**current, previous, nextt**) to reverse the list without any significant extra memory usage, resulting in **constant** **space** complexity, O(1).



**Doubly Linked List**



**Representation of Doubly Linked List in Data Structure**

In a data structure, a doubly linked list is represented using nodes that have three fields:

1. Data
2. A pointer to the next node (**next**)
3. A pointer to the previous node (**prev**)

