Find the time complexity involved with following operations on the singly linked list:

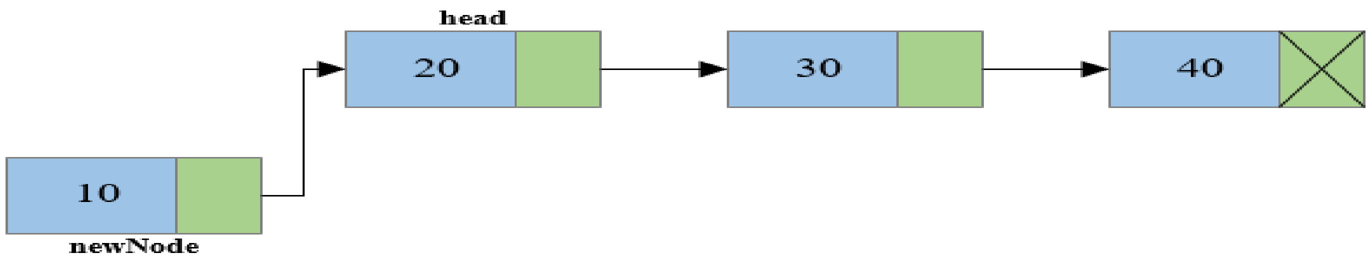
Time Complexity of singly linked list:

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Singly Linked List	θ(n)	θ(n)	θ(1)	θ(1)	O(n)	O(n)	O(1)	O(1)	O (n)

1. Insert and delete a node at the start of the list:

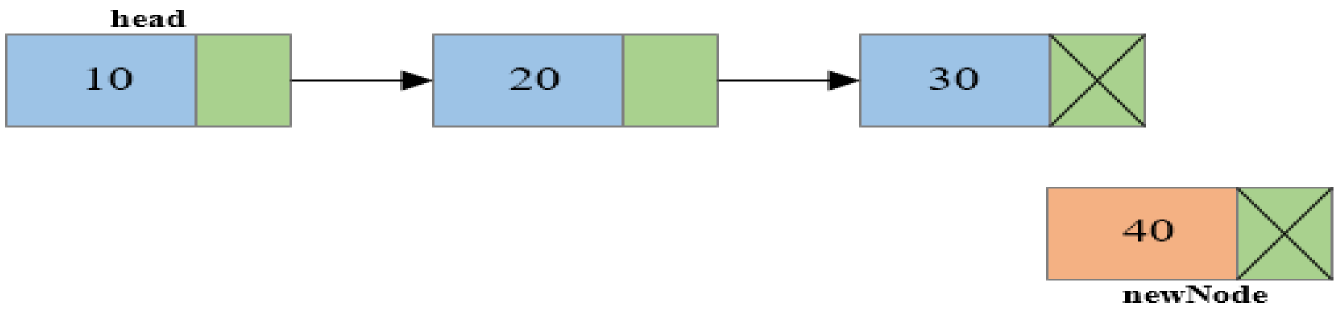
- **Insert a node at the start of the list:**

It involves inserting any element at the front of the list. We just need to a few link adjustments to make the new node as the head of the list.



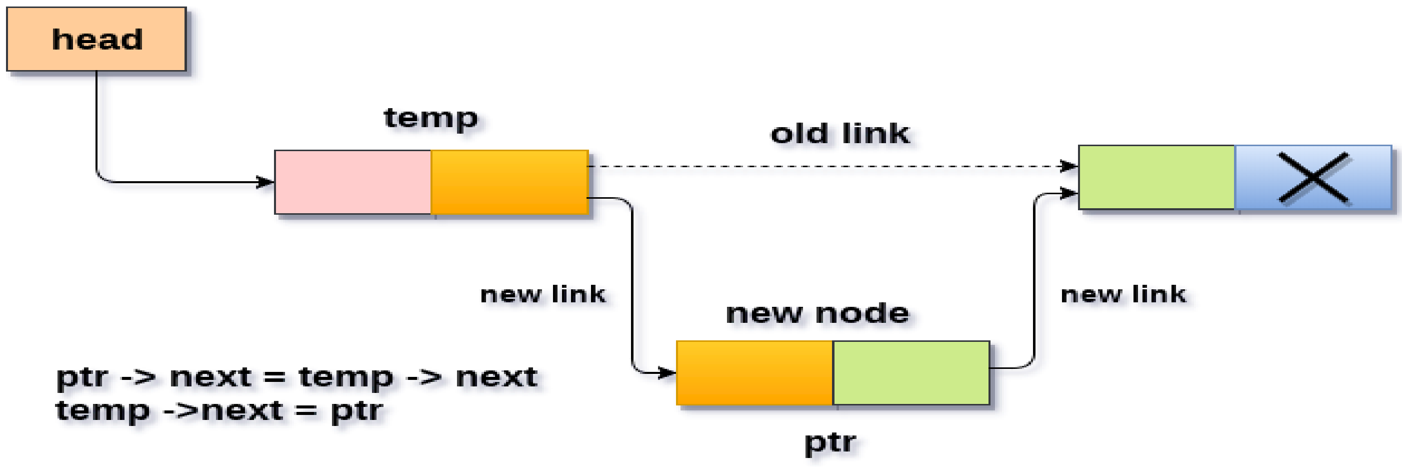
- **Insert a node at the end of the list:**

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.



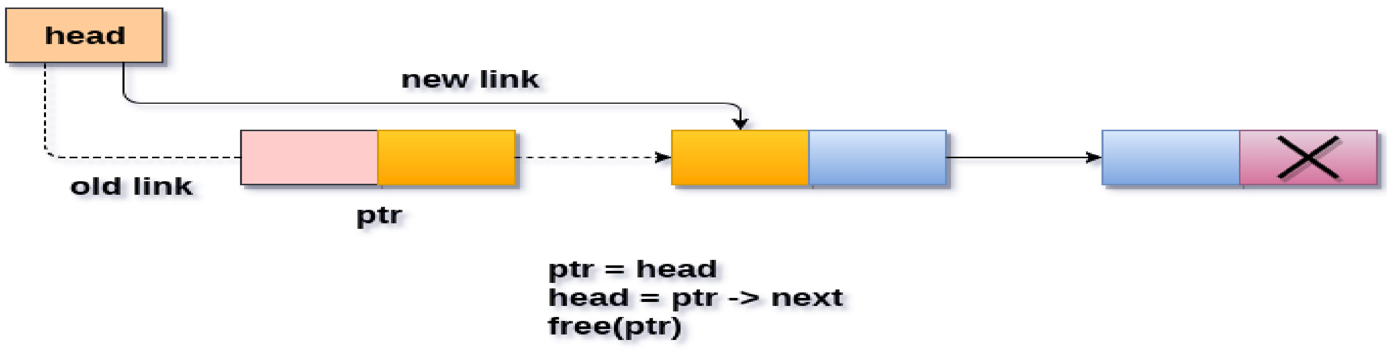
- **<u>Insert a node after a given node:</u>**

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.



- **<u>Delete a node at the start of the list:</u>**

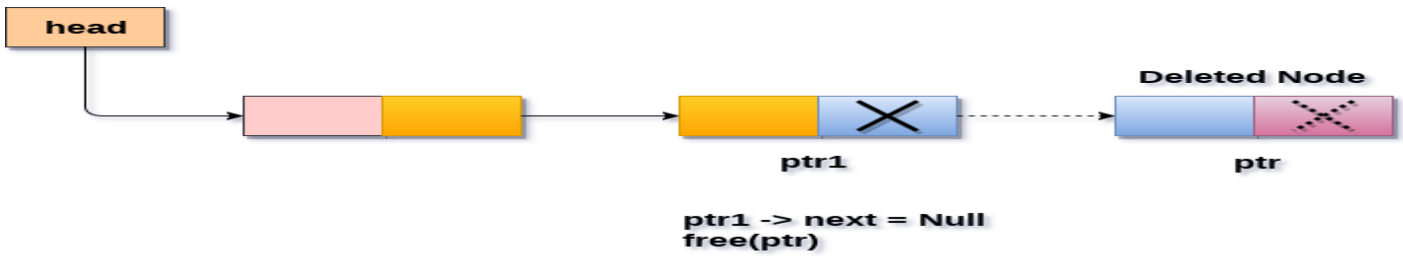
It involves deletion of a node from the beginning of the list. This is the simplest operation among all. It just needs a few adjustments in the node pointers.



Deleting a node from the beginning

- **<u>Delete a node at the end of the list:</u>**

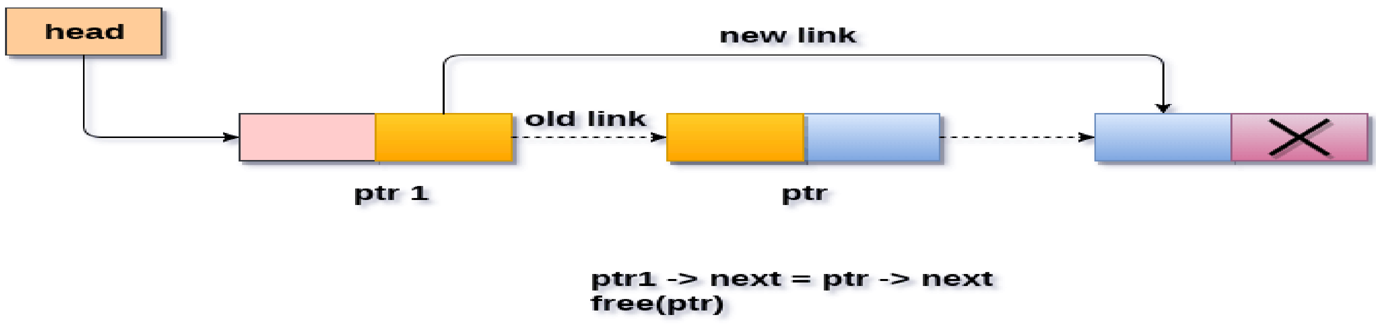
It involves deleting the last node of the list. The list can either be empty or full. Different logics are implemented in each scenario.



Deleting a node from the last

- **Delete a node after a given node:**

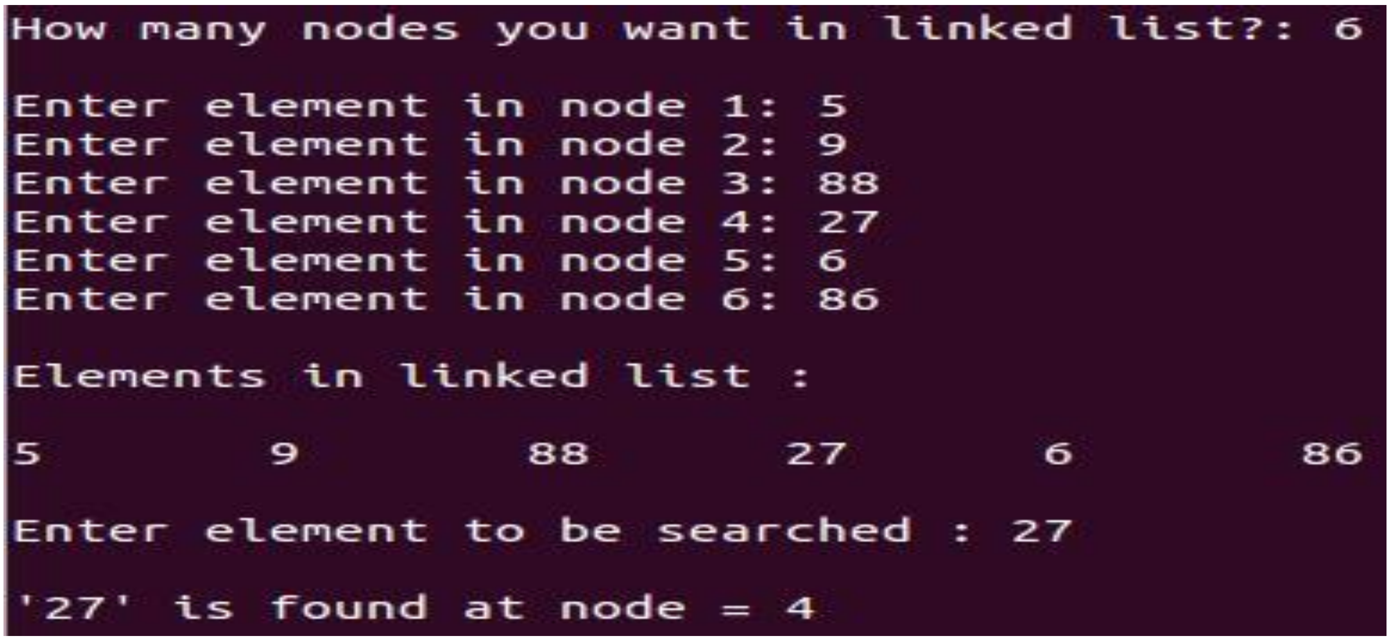
It involves deleting the node after the specified node in the list. we need to skip the desired number of nodes to reach the node after which the node will be deleted. This requires traversing through the list.



Deletion a node from specified position

2. Search an element in the list:

In searching, we match each element of the list with the given element. If the element is found on any of the location, then location of that element is returned otherwise null is returned.



```
How many nodes you want in linked list?: 6

Enter element in node 1: 5
Enter element in node 2: 9
Enter element in node 3: 88
Enter element in node 4: 27
Enter element in node 5: 6
Enter element in node 6: 86

Elements in linked list :

5        9        88        27        6        86

Enter element to be searched : 27

'27' is found at node = 4
```