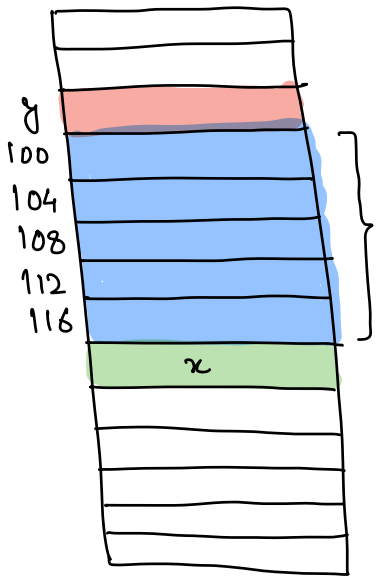
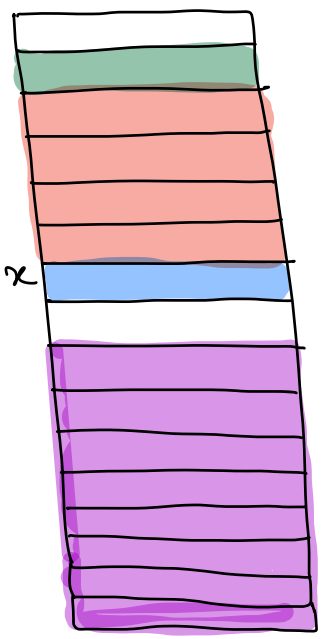
int a[5];

↓

~~4B~~

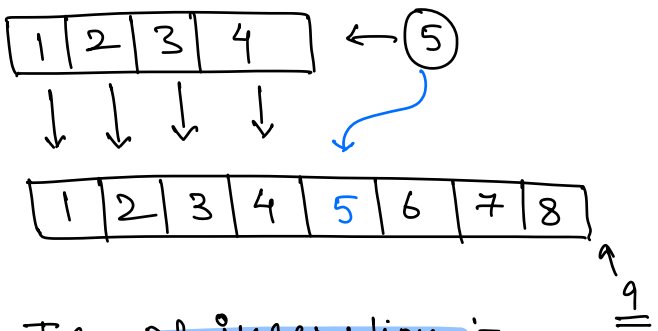
(Static)





Dynamic Arrays

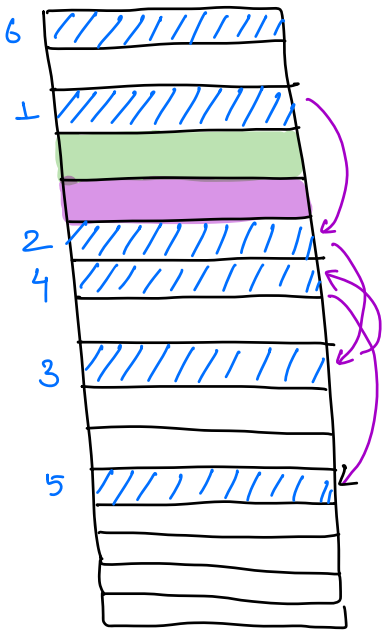
vector / ArrayList

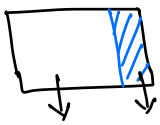


TC of insertion :-

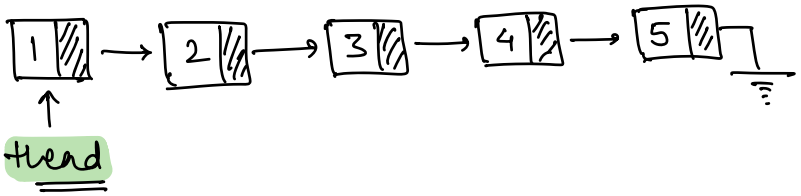
Worst case : O(N)

Amortized TC : O(1)





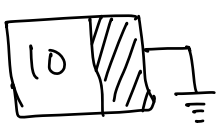
data reference of the next node.



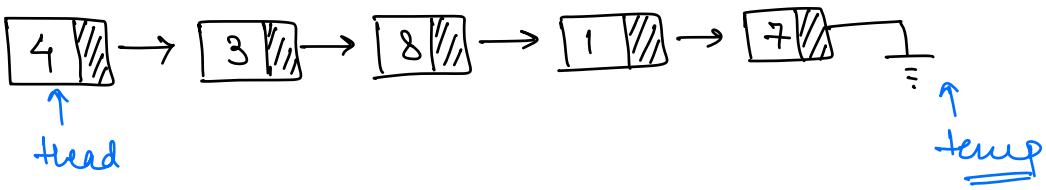

```
Class Node {
    int data;
    Node next;
    Node (int x) {
        this.data = x;
        this.next = Null;
    }
}
```

```
Node n = new Node()
n.data ?
n.next ?
```

```
Node n = new Node(10);
```



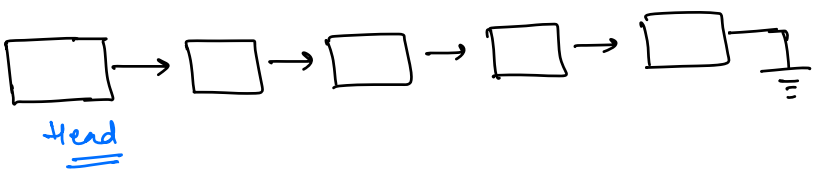
Q: Given a L.L, find its length.
Head is given



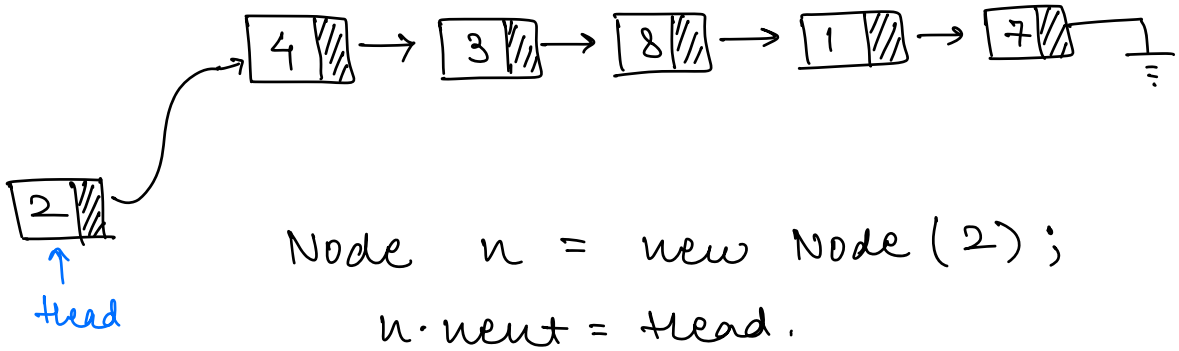
l = 0 1 2 3 4 5

```
int length(Node head){
    l = 0;
    Node temp = head;
    while(temp != Null){
        l++;
        temp = temp.next;
    }
    return l;
}
```

* Insertion in L.L



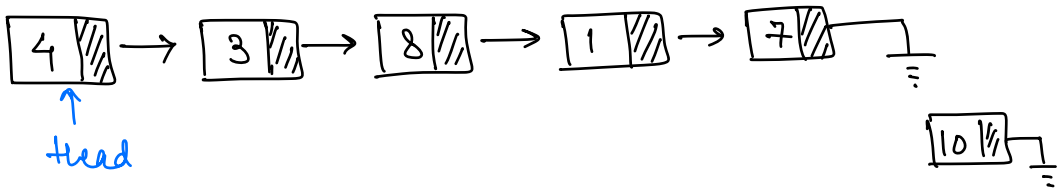
1) Insert at Head / Start :-




```
Node n = new Node(2);
n.next = head.
head = n;
```

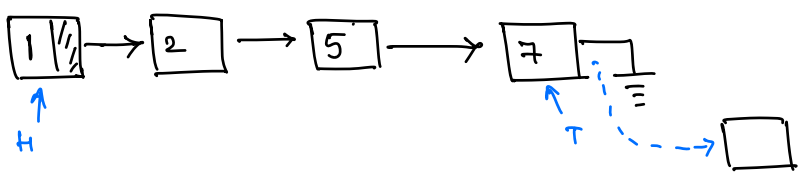
TC : O(1)

2) Insert at End :-



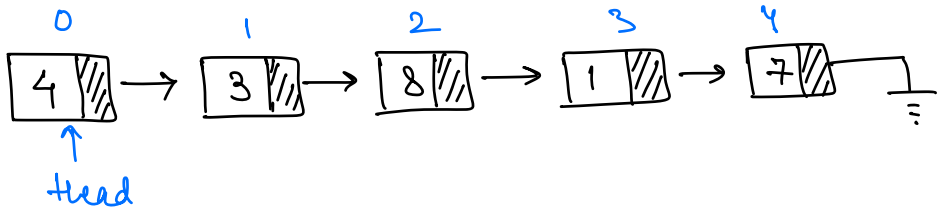
⇒ TC : O(N)

* Optimisation :- Maintain tail pointer as well.

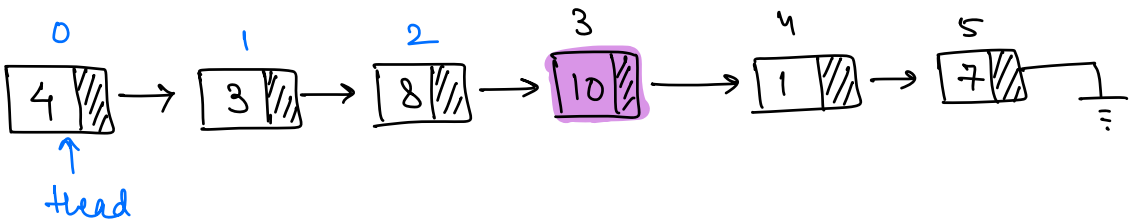


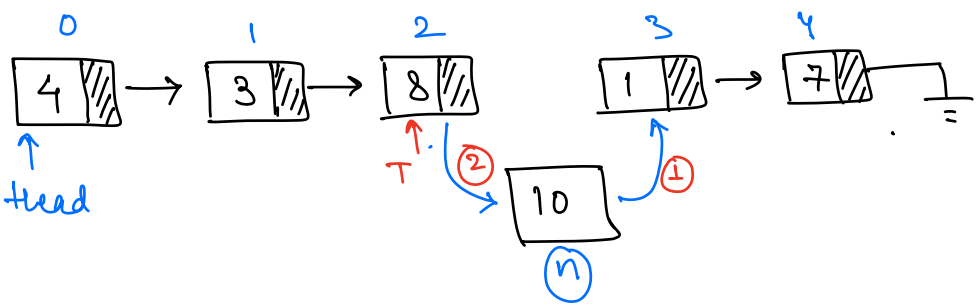
TC : O(1)

3) Insert at K^{th} position :-



K = 3



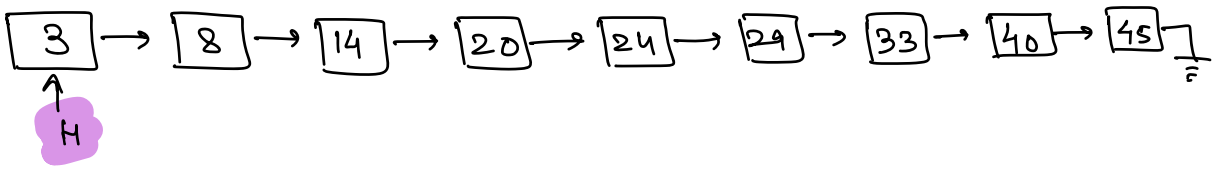


TC : O(N)

Edge cases :-

1) Head is Null.
2) L.L of size = 1 | 2 | 3.
3) Insertion | Deletion.
4) Problem Specific edge cases.
   (Ex :- for above problem :
   K = 0 | K > N)

Q: Given a L.L sorted in ascending order, Insert a value at its correct position maintaining the sorted order.



K = 23

K = 50

* find the last node with value less than K.

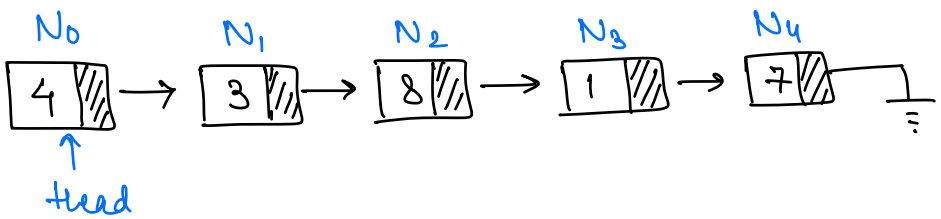
```
Node insertInSortedOrder (head, K) {
    Node n = new Node(K);
    // If head is NULL
    if(head == NULL) {
        return n;
    }
    // If the node is being inserted at start.
    if(K < head.data) {
        n.next = head;
        return n;
    }
    // Traverse the L.L and find the
    // correct pos^ for new node.
    Node temp = head;
    while (temp.next != Null && temp.next.data < K) {
```

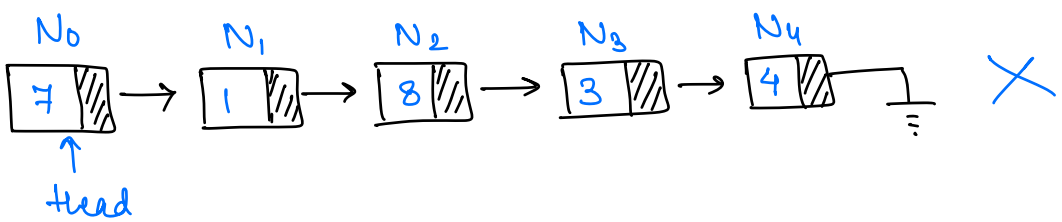
```
        temp = temp.next;
    }
    n.next = temp.next
    temp.next = n;
    return head;
}
```

TC: O(N)

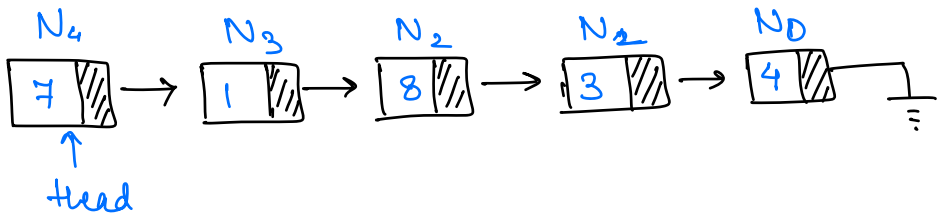
Q: Reverse the L.L

- SC: O(1)
- Changing the value of a node is NOT allowed.

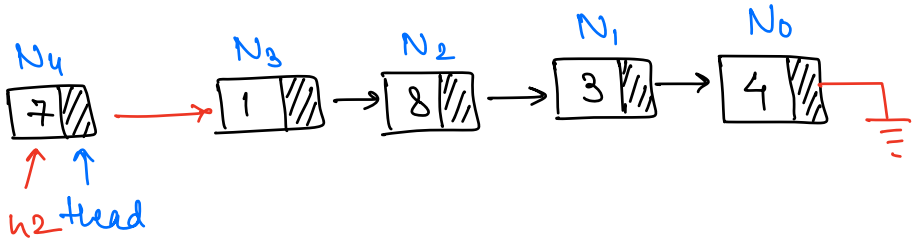




✗



✓

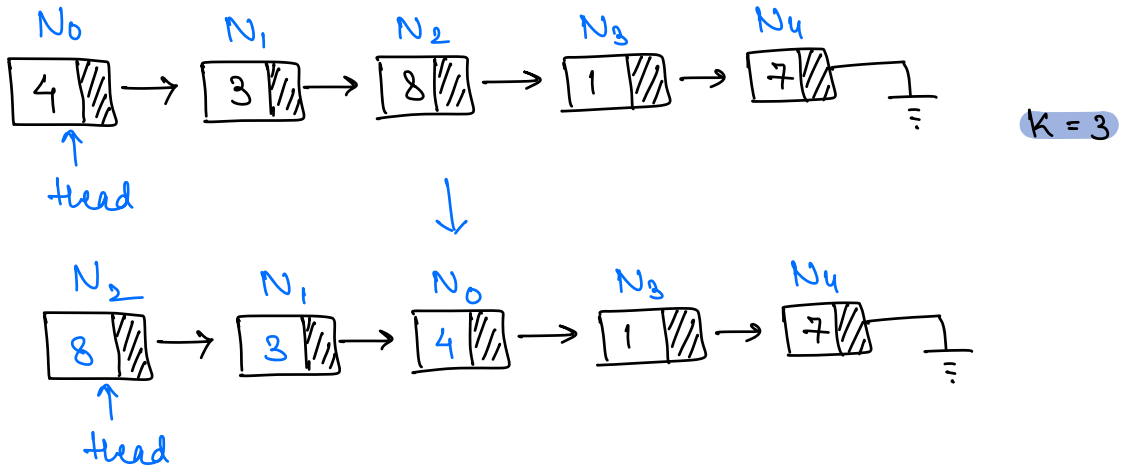



```
Node reverse (Node head) {
    h1 = head, h2 = Null;
    while (h1 != Null) {
        temp = h1;
        h1 = h1.next;
        temp.next = h2;
        h2 = temp.
    }
    return h2;
}
```

TC : O(N)

SC : O(1)

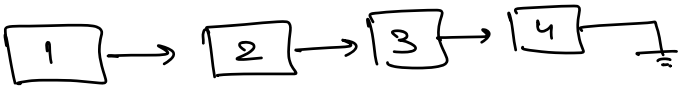
Q: Given a L.L, Reverse the first (K) nodes of the L.L. (SC: O(1))



* K > N ⇒ Reverse the complete L.L

```
Node reverseFirstK ( head , K) {
  if( K == 0 || head == Null) return head;
  h1 = head , h2 = Null;
  while( h1 != Null && K > 0) {
        temp = h1;
        h1 = h1.next;
        temp.next = h2
        h2 = temp
        K--;
  }
  ⇒ head.next = H1;
    return h2;
}
```

Doubts



K = 6

K > N X

K > N