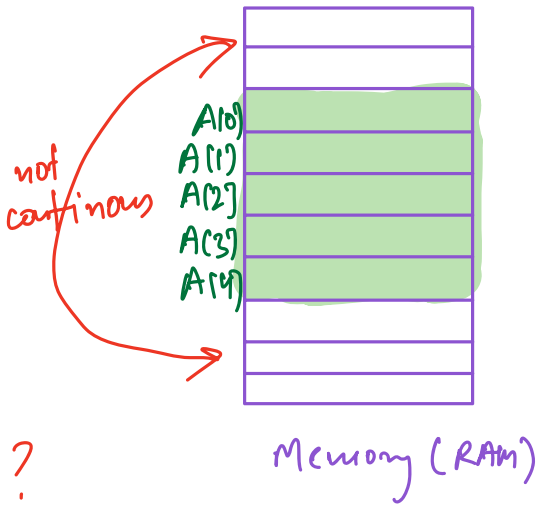
Linked list Basics

Array : O(1) random access

A[i] → O(1) TC

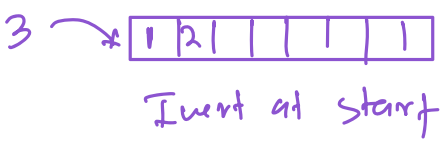
int A[5] → how it will be stored ?



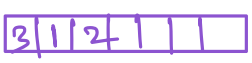
can you store B[4] in the memory ?

↳ Can't create since we don't have 4 contigous locations. [Memory fragmentation]

	At Start	At end	At K^{th} position (random)
Insertion	O(N)	O(1)	O(N)
Deletion	O(N)	O(1)	O(N)



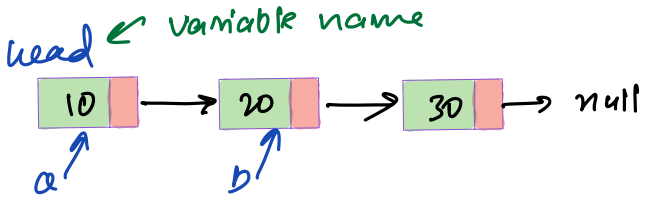
⇒



Shift existing elements to right & then insert

Linked List

```
class Node {
    int data;
    Node next;
    Node(int val) {
      self.data = val
      self.next = null
    }
}
```




```
Node a = new Node(10)
Node b = new Node(20)
a.next = b
b.next = new Node(30)
```

Create a list of 4 values → A[4]

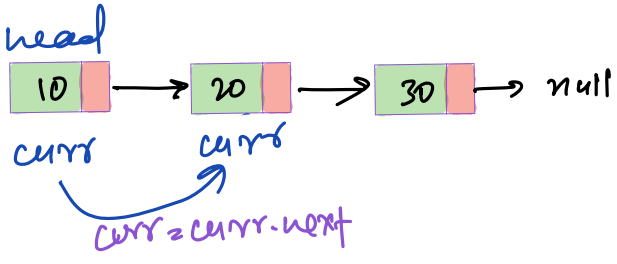
```
Node head = new Node(A[0])
head.next = new Node(A[1])
head.next.next = new Node(A[2])
head.next.next.next = new Node(A[3])
```

Print

```
head.data
head.next.data
head.next.next.data
:
:
:
```

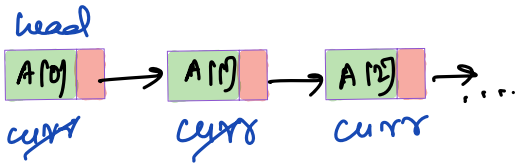
Iterate over linked list

```
Node curr = head
while(curr != null) {
    print(curr.data)
    curr = curr.next
}
```



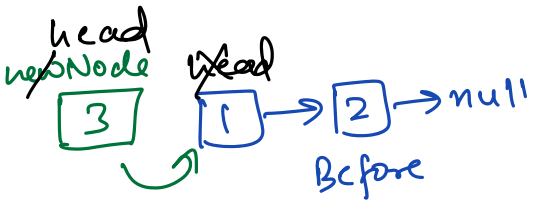
⇒ 10 20 30

Create LL using loop

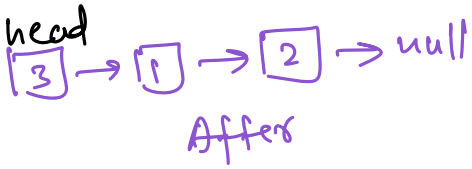



```
Node head = new Node(A[0])
    curr = head
for (i=1; i<n; ++i) {
    curr.next = new Node(A[i])
    curr = curr.next
}
```

Insert At Start

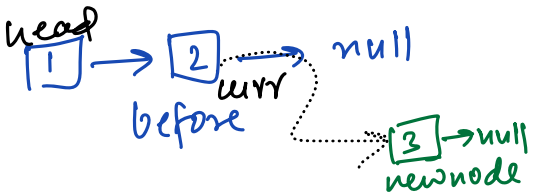



```
Node insertAtStart(head, val) {
    Node newNode = new Node(val)
    newNode.next = head
    head = newNode
    return head
}
```

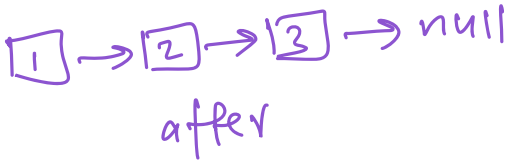


TC : O(1)

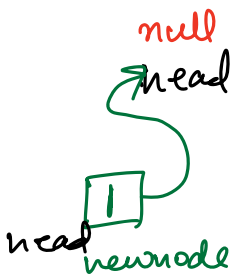
Insert at end



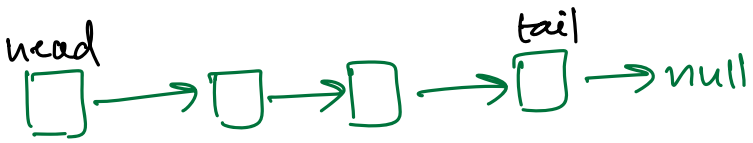

```
Node insertAtEnd(head, val) {

    Node newNode = new Node(val)

    if (head == null)
        return newNode

    Node curr = head
    while(curr.next != null) {
        curr = curr.next
    }
    curr.next = newNode
    return head
}
```



fail if head is null

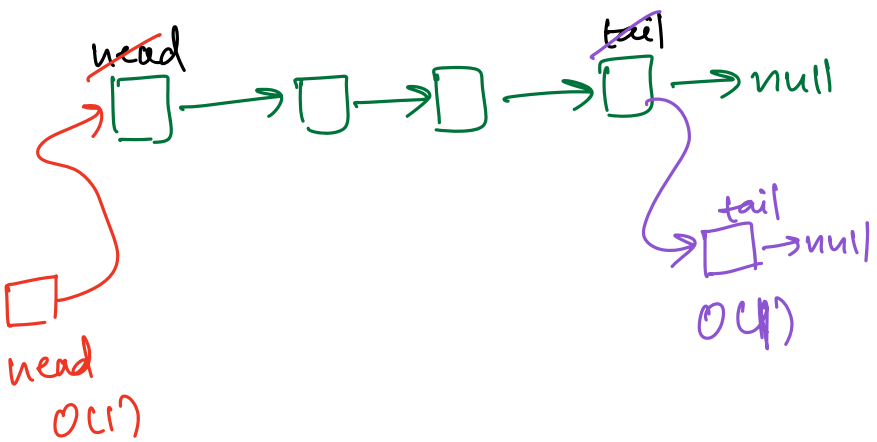


TC: O(N)

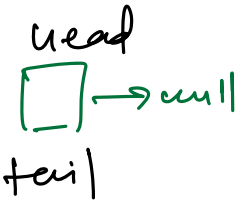


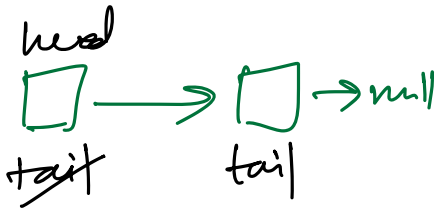
Using tail reference we can get O(1)

```
tail.next = newNode
tail = newNode
```

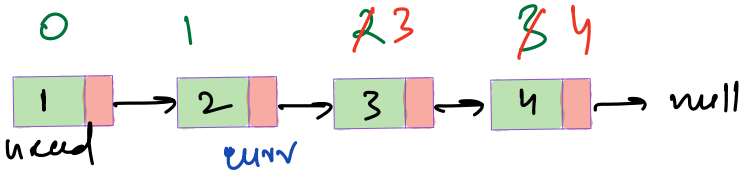


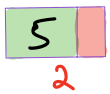

```
head = null
tail = null
```





Insert at K^{th} position (random)





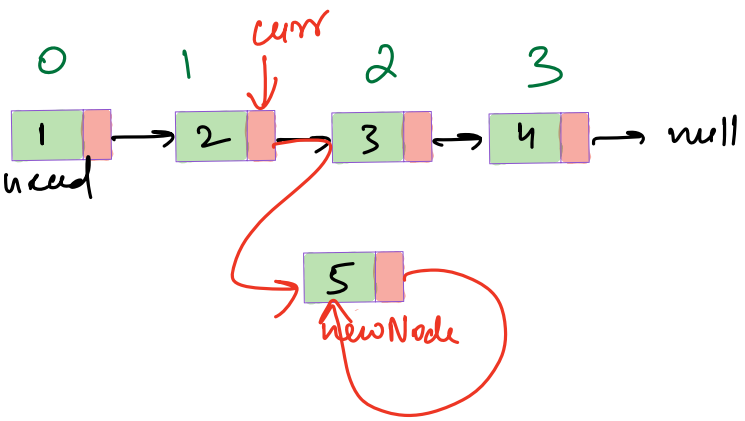
insert at K=2

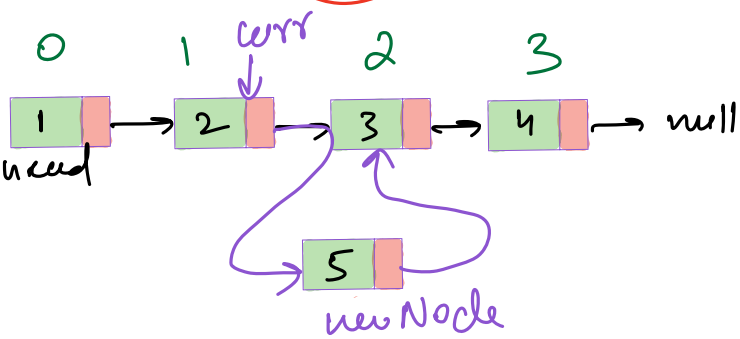
to reach curr, iterate K-1 times

```
Node insertAtK(node, val, K) {
  newNode = new Node(val)
  curr = head
  for(i=0; i<K-1; ++i) {
      curr = curr.next
  }
```

✗
```
  curr.next    = newNode
  newNode.next = curr.next
```

✓
```
  newNode.next = curr.next
  curr.next    = newNode

  return head
}
```

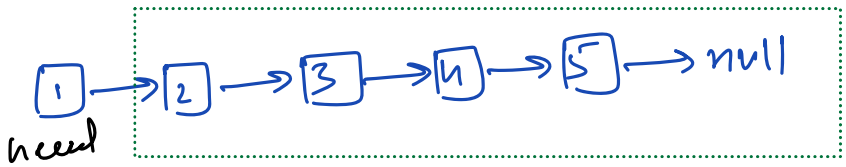




TC: O(K) ⇒ O(N)

Question

Print LL in reverse order.



Print (head)

main question

⇓

5 4 3 2 1

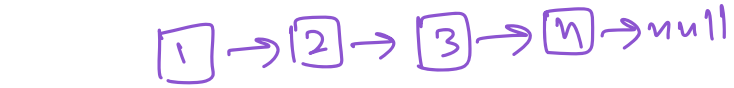
Print (head.next)

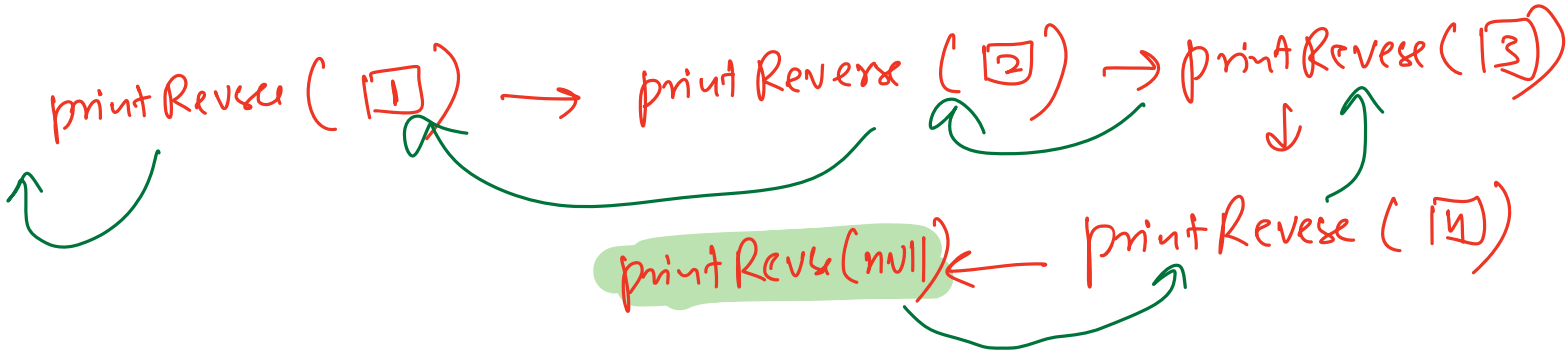
⇓

5 4 3 2

printReverse (head)

```
void printReverse (Node curr) {
    if (curr == null) { return }
    printReverse (curr.next)
    print (curr.data)
}
```





Print: 4 3 2 1

Recommendations

1. Write code on paper
2. Dry run
3. Never use "hit & trial"
4. Edge Cases → LL is null? (head = null)
   Size of LL = 1?
   size = 2/3?
   problem specific