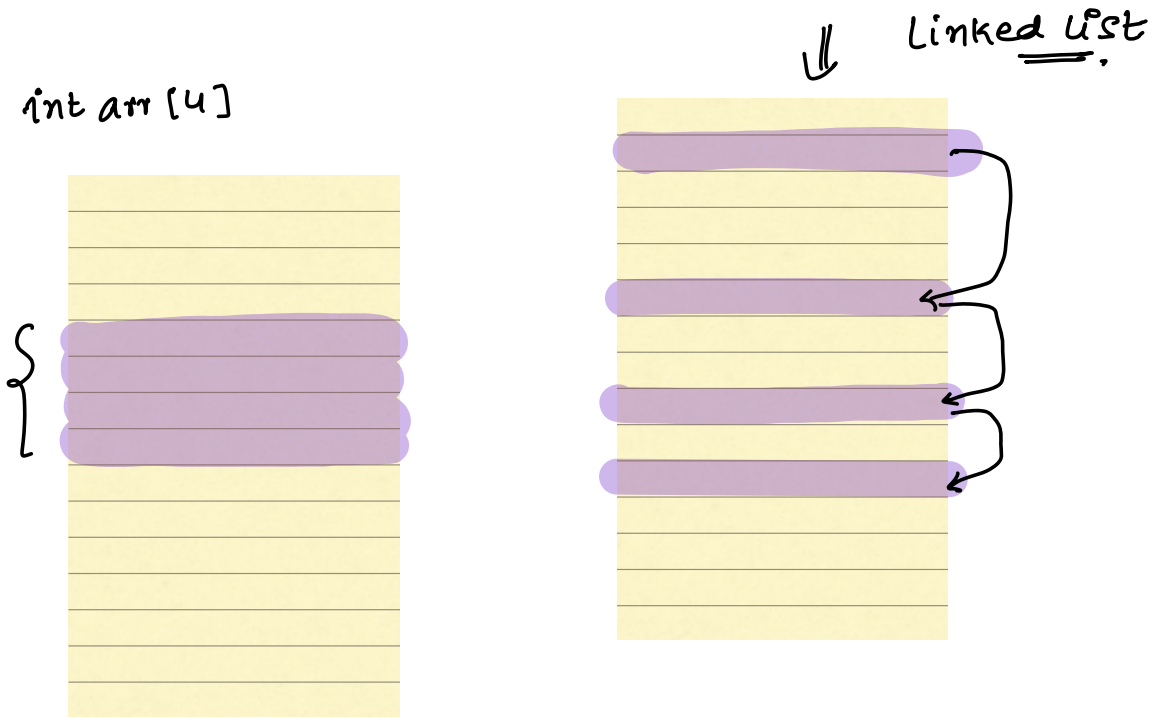
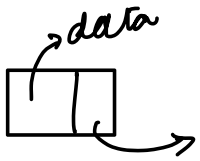
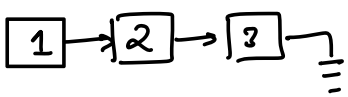
→ Linked list Basics
→ Size
→ Get Kth pos ele
→ insert (Node head, int pos, int val)
→ Delete (Node head, int pos)
→ Delete allOccurrence ( )
→ Reverse K Nodes
→ Reverse K groups.



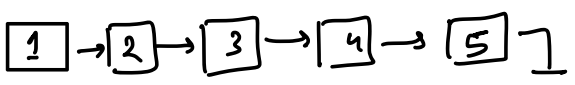
```
class Node {
    int data
    Node next

    Node (int x){
        data = x
        next = NULL
    }
}
```

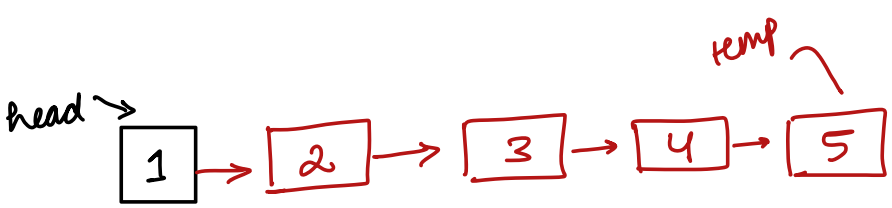


N = 3



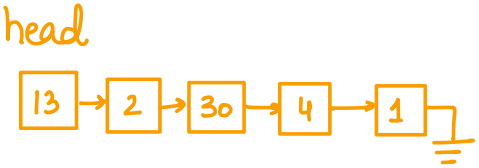
N=5






```
Node head = new Node(1)
Node temp = head
for( i=2 ; i<= N; i++){
    temp.next = new Node(i)
    temp = temp.next
}
return head .
```

Q2. Given a LL, calc size (head)



ans → 5

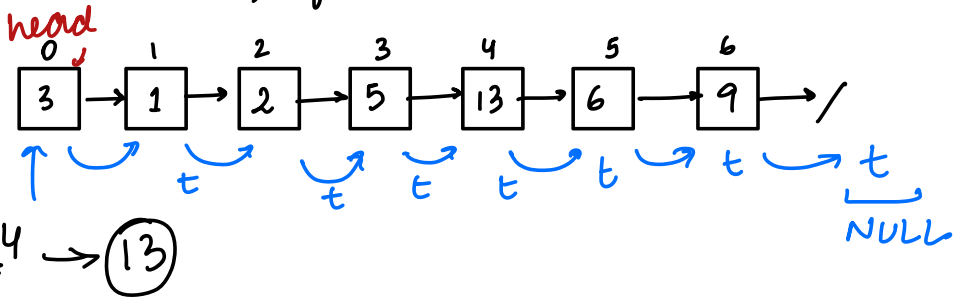
cnt = 0 1 2 3 4 5

t ↑

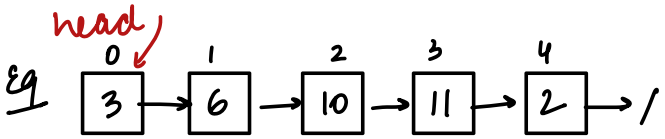
```
Node t = head
cnt = 0
while ( t != NULL) {
    cnt ++
    t = t.next
}
return count.
```

(-1) =

Que. Given a LL, find data at Kth pos [0 based ind]

Eg 

K = 4 → (13)



K = 3 → (11)

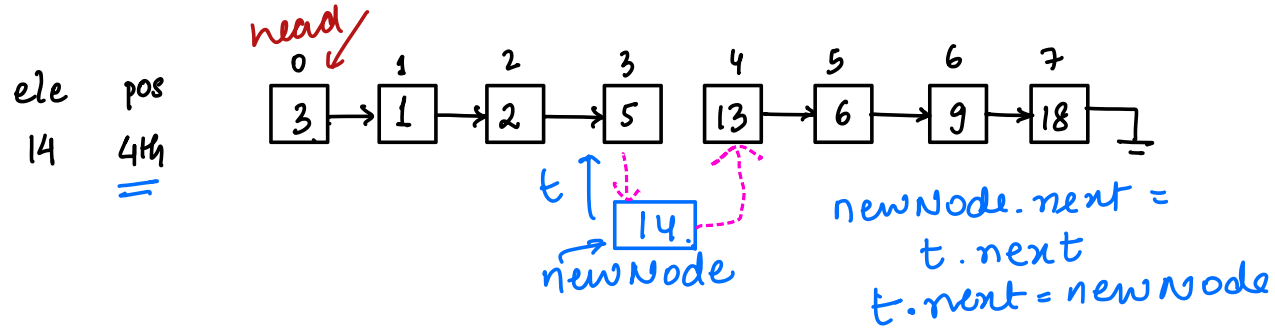
```
int getKthNode (Node head, int K){
    Node t = head
    while (K>0 && t!=NULL){
        t = t.next

        K--
    }
    if (t!=NULL) return t.data
    else return -1
}
```

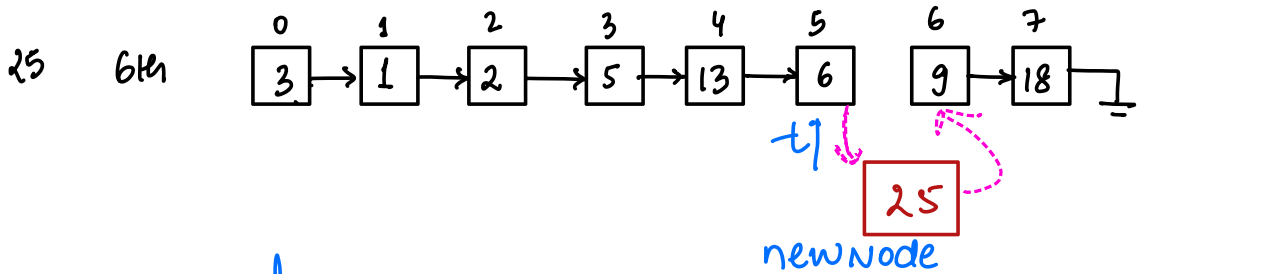
K = 0 ✓
K >= size ✓
head == NULL ?

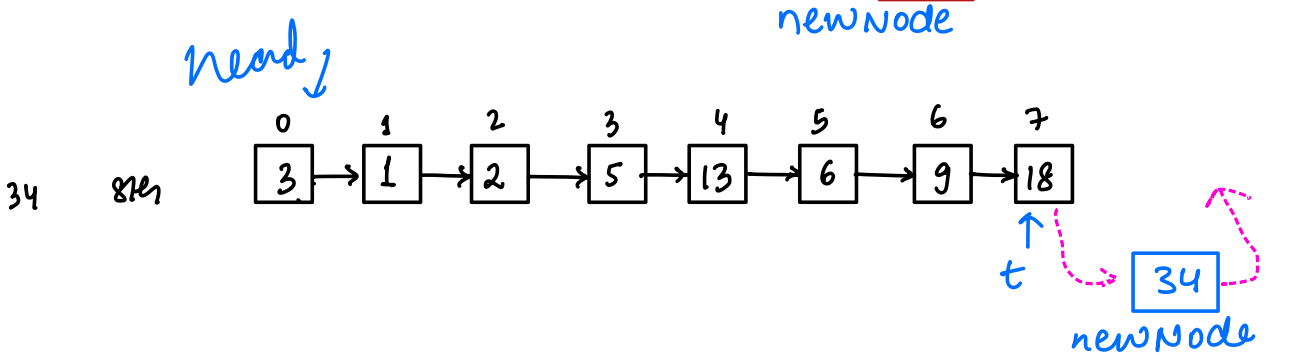
Insert

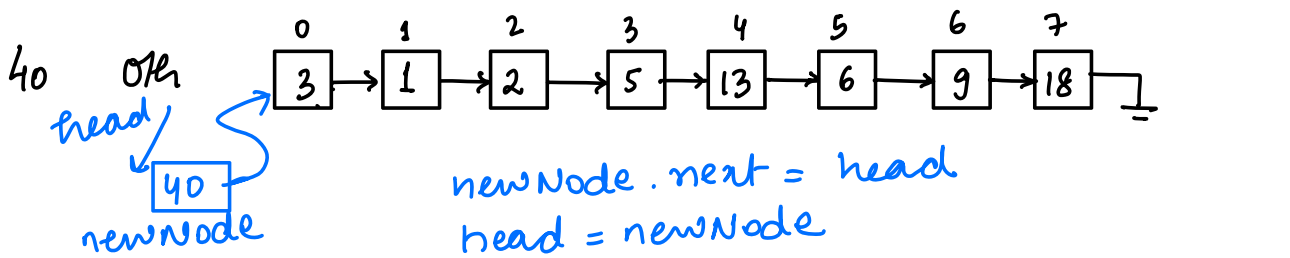
Insert an ele at a pos

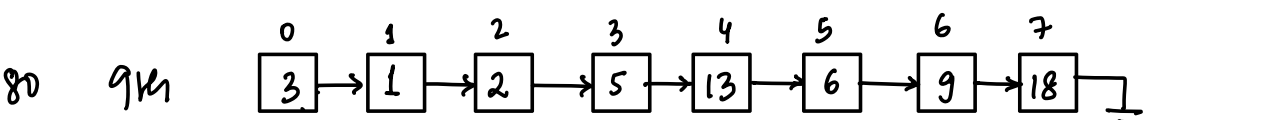
Node newNode = new Node(14)











* Not possible

```
Node insert (Node head, int k, int ele){
    //not possible
    if (k > size || k < 0) {
        return head
    }
    Node newNode = new Node (ele)
    // insert at the head
    if (k == 0){
        newNode.next = head
        head = newNode
        return head
    }
    //insert in b/w
    Node t = head
    for(i = 1 ; i < k ; i++){
        t = t.next
    }
    newNode.next = t.next
    t.next = newNode
    return head
}
```

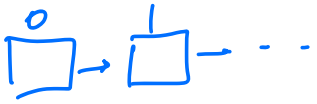
write the size func

head = NULL
k = 0

head = NULL
k > 0

head
↓
NULL

k = 2



TC: O(N+N)

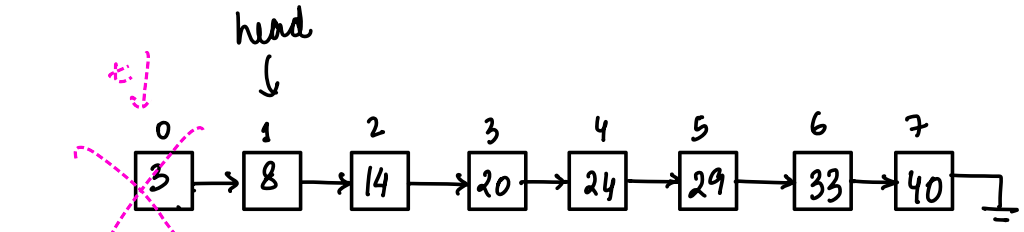
O(N)

SC: O(1)

Break: 8 min
10:33

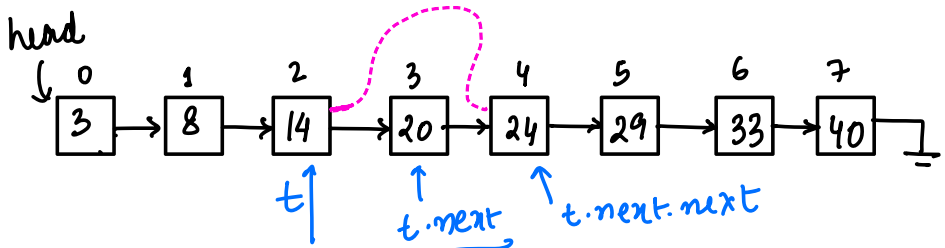
Deletion

K = 0



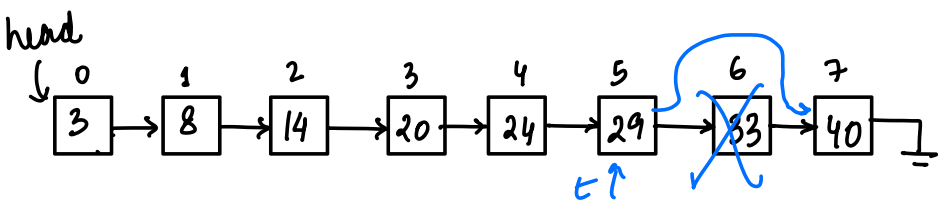
t = head
delete(t)
head = head.next

K = 3



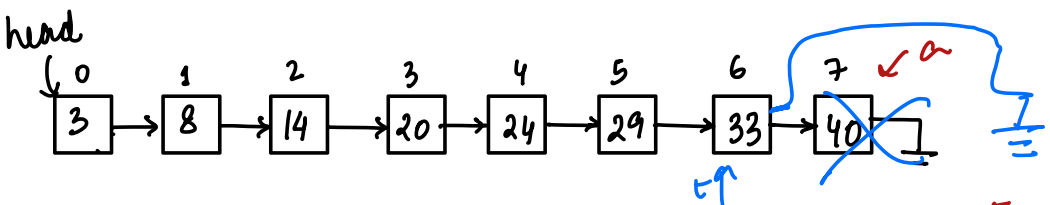
a = t.next
t.next = t.next.next
delete (a)

K = 6



a = t.next
t.next = t.next.next
delete (a)

K = 7



a = t.next
t.next = t.next.next

null.next ✗
null.data ✗

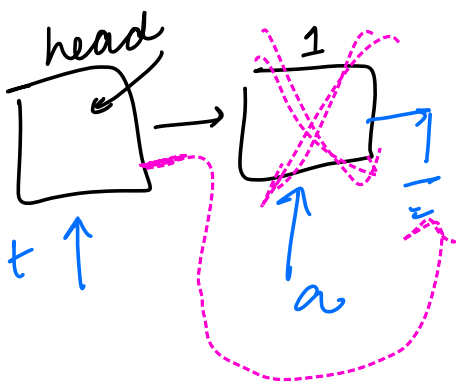
```
Node deleteLL ( Node head , int k){
    //invalid case
    if(k >= size || k<0){
    |  return head
    }
    //valid cases.
    //Delete from head
    if (k==0){
    |   Node t = head
    |   head = head.next
    |   delete (t)
    |   return head
    }
    //Delete from anywhere else
    Node t = head
    for( i=1 ; i<k ; i++){
    |      t = t.next
    }
    a = t.next
    t.next = t.next.next
    delete (a)
    return head
}
```

head = NULL

k >= size

Single Node

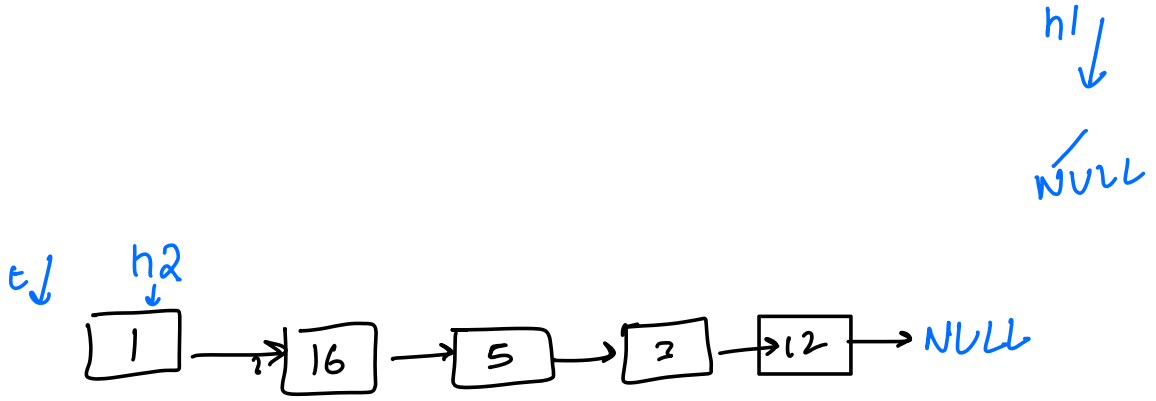
Two nodes



Reverse Linked List

many companies

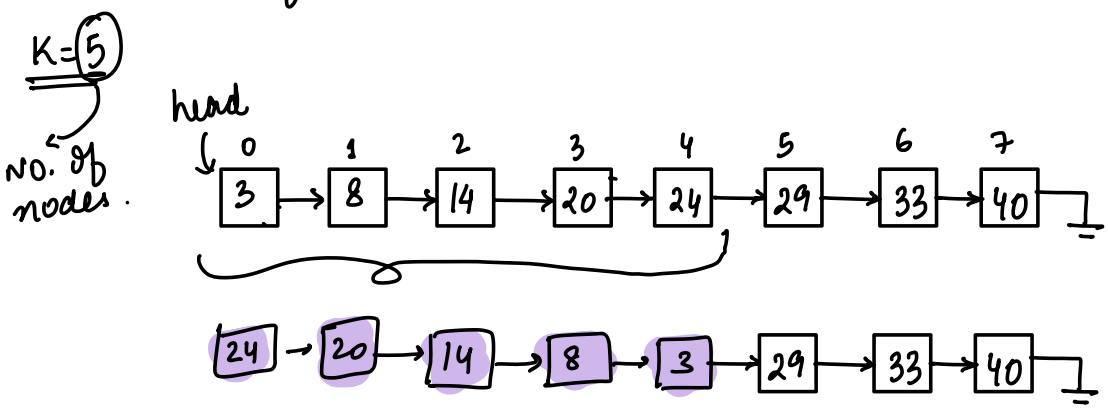
* No extra space
* Cannot modify the data on the node




```
Node reverse (Node head){
    h1 = head
    h2 = NULL
    while (h1 != NULL) {
        t = h1
        h1 = h1.next
        t.next = h2
        h2 = t
    }
    return h2
}
```

TC : O(N)

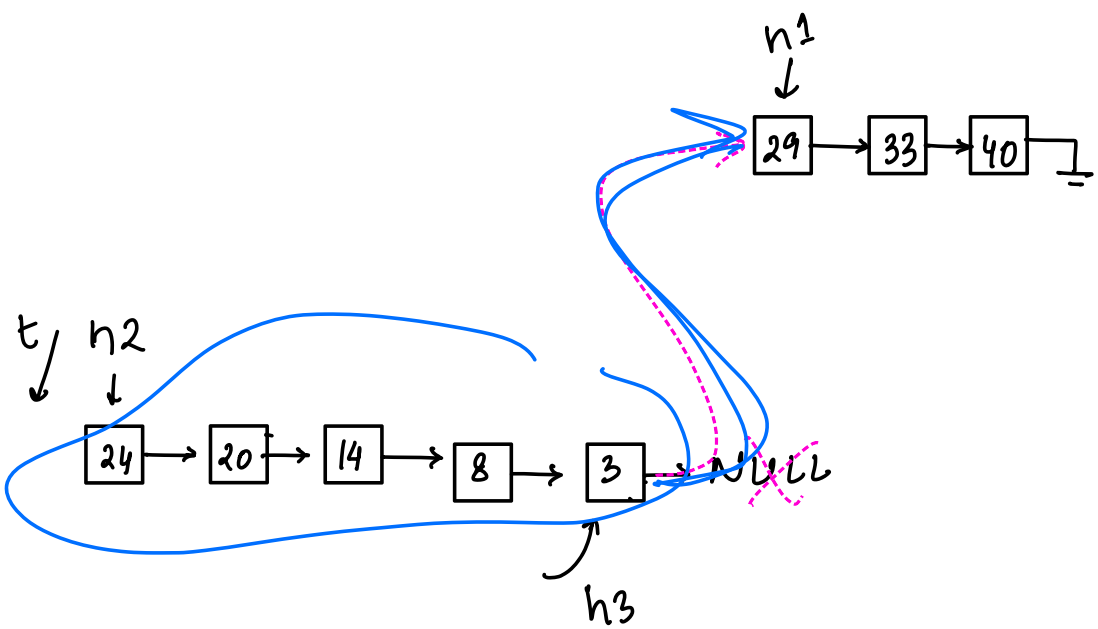
Qu. Reverse first K nodes of linked list.



if K>=N : Reverse the entire linked list

h1 = t = h3 = head

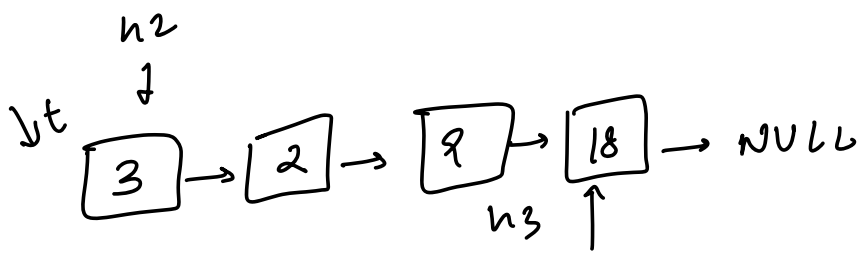
K = 5



K h1

h1
NULL

K = 5



h1 = h3 = t = head

h2 = NULL

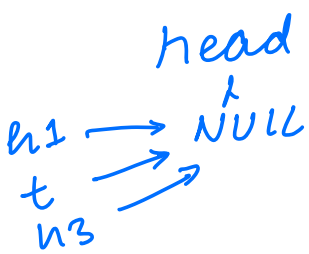
while (K>0 && h1 != NULL) {

    t = h1
    h1 = h1.next
    t.next = h2
    h2 = t
    K--

}

if(h3 != NULL) { h3.next = h1 }

return h2



h2
NULL

Que. Reverse LL in group of K

K = 4