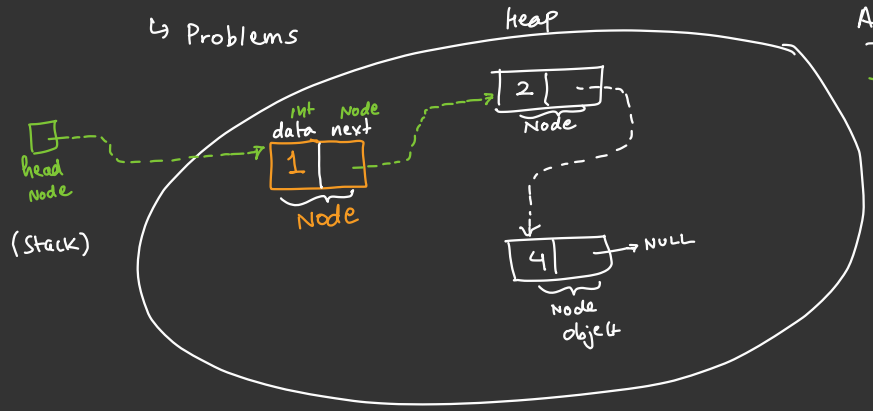
LINKED LIST - (1)

11 Sept - Extra class on Hashing (New Problems)

↳ Recap

↳ Problems



Advantage →

→ grow & shrink acc to demand.

→ insert at or index if O(1)

SDE → good news ⇒ Library LinkedList

LinkedList l

→ l.insert(0,5)
l.insert(2,8)
l.remove(8)

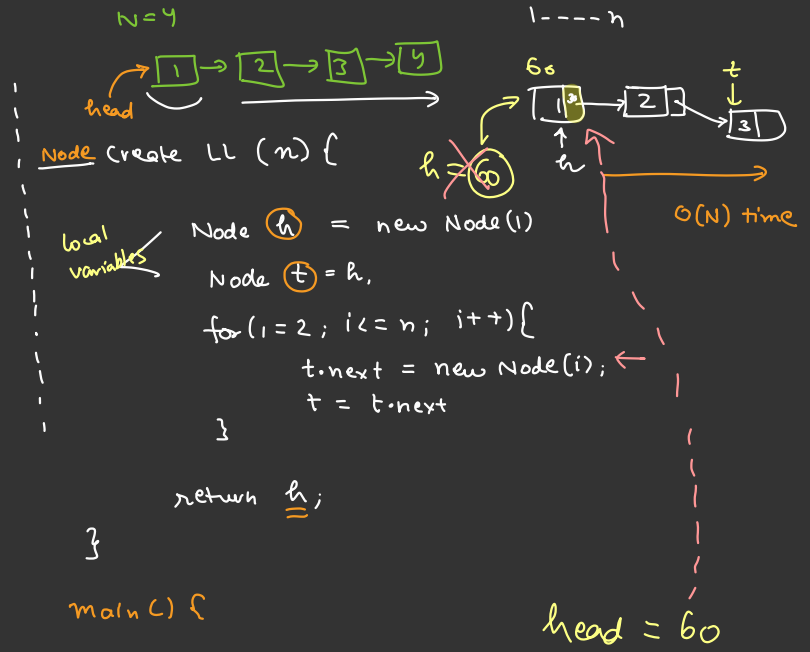
bad News ⇒ learn how to manipulate/update a LinkedList

↓

- Interviews
- good understanding

Java Implemenation

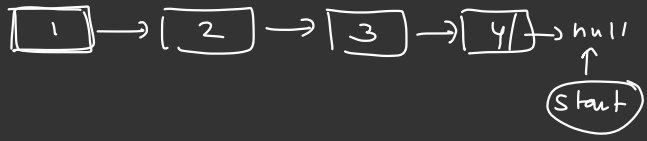
```
class Node {
    int data
    Node next,
    Node (int x) {
        data = x
        next = NULL
    }
}
```



local variable

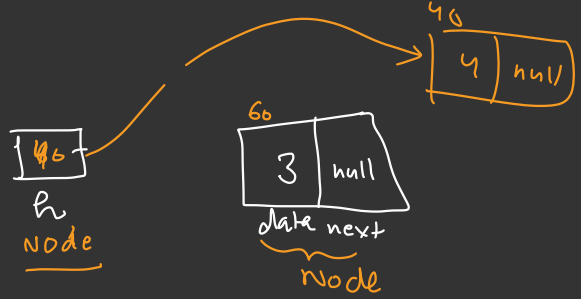
Copy of head

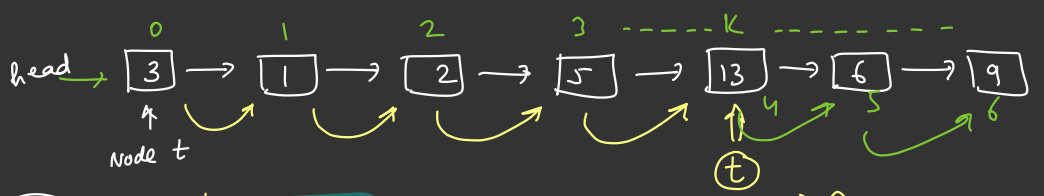
```
int Size ( Node start ) {
    Cnt = 0
    while ( start != null ) {
        cnt += 1
        start = start.next;
    }
    3
    return cnt
}
```



~~1 2 3~~ 4

```
Node h = new Node(3);
```





int getKthElement(Node head, int k){

Node t = head,

if(k >= size(head)) { return -1 };

while(k > 0){

t = t.next;

k--

}

return t.data,

}

K >= 0

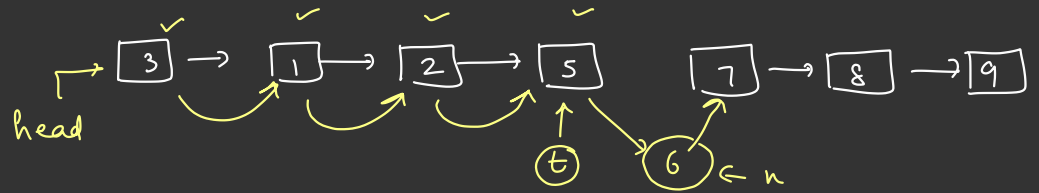
K = 0

K > size(head).

K = 6

→ size = 0 K = 0

→ head == null



K = 4

Node insert (Node head, int K, int data)

K = 4 ⇒ after K nodes start with 1

data = 6

```
if( K > size(head)){
```

Case-I

```
return head
}
```

```
if( K == 0){
```

Case-II

```
Node n = new Node(data);
n.next = head;
return n;
}
```

```
main(){
head = insert(head, 0, 0)
}
```

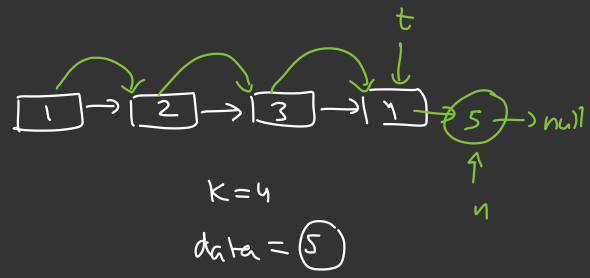
Case-III

Travel

```
Node t = head
for(i=1; i<= K-1; i++){
    t = t.next
}
```

middle / end

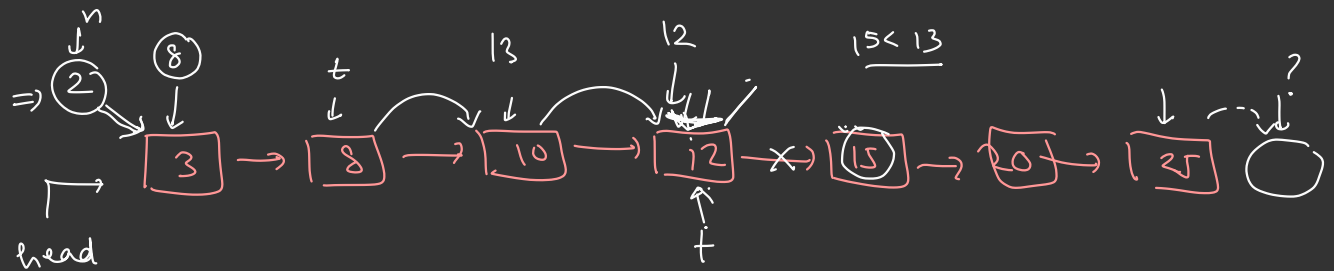
```
insert {
    Node n = new Node(data);
    n.next = t.next
    t.next = n;
    return head;
}
```



head = insert(head, 2, 25);
100 100

head = insert(head, 0, 0);
220 100

(6) Given a sorted linked list, insert element at its correct pos



→ Insert (13)

→ Insert (2)

TODO

10 14

Node insertSorted(head, data) {

⇒ if(head==null || data < head.data){

Node n = new node(data)

n.next = head;

return n;

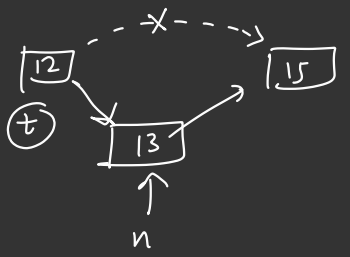
3

Node t = head;

⇒ while (t.next!=null && t.next.data < data){

t = t.next

① ②

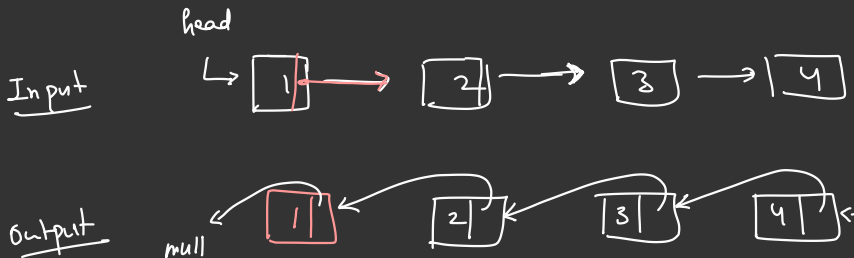



```
    }
    Node n = new Node(data);
        n.next = t.next;
        t.next = n
        return head;
}
```

⇒ Reverse Entire Linked List, on same list.

10.45

15 min break :)



All Nodes
→ (Save the next Node first)
→ current → next = prev ;
→ Track the prev Node

prev
current
n

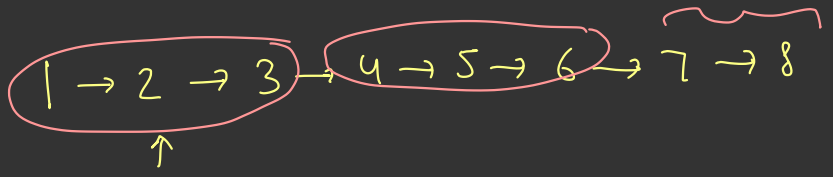
```
Node reverse (head) {
    prev = null
    current = head
    while (current != null) {
        n = current → next
        current → next = prev
        prev = current
        current = n
    }
    return prev;
}
```

Save
Updating the link
} update prev & current

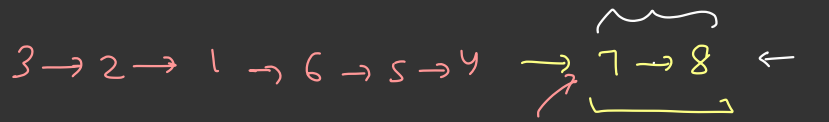
☆ Interview Problem

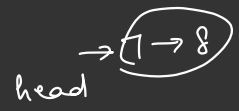
3

⇒ Reverse a LL in groups of size K.



K = 3

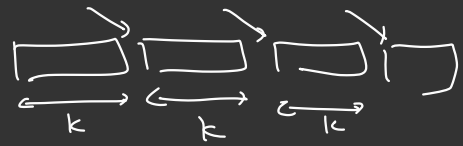





```
Node reverseK(head, K) {

    // Base Case
    if( size(head) < K ){        // length of LL
        return head;
    }

    // Rec Case
```



$\left(\frac{N}{K}\right)$ calls × K

$= O(N)$

[LL of Size K]

→ P = null

→ C = head , cnt = 0

while (cnt < K){

N = C.next

C.next = P

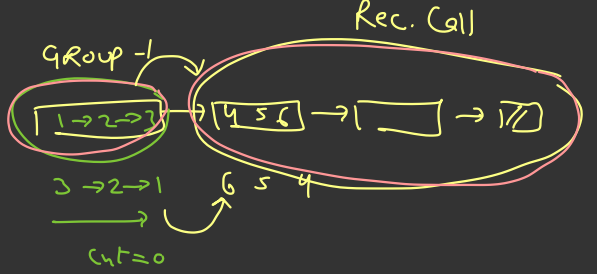
P = C

C = N

cnt++

}

{



Rec. Call

if (current != null){

head.next = reversek (current , k);

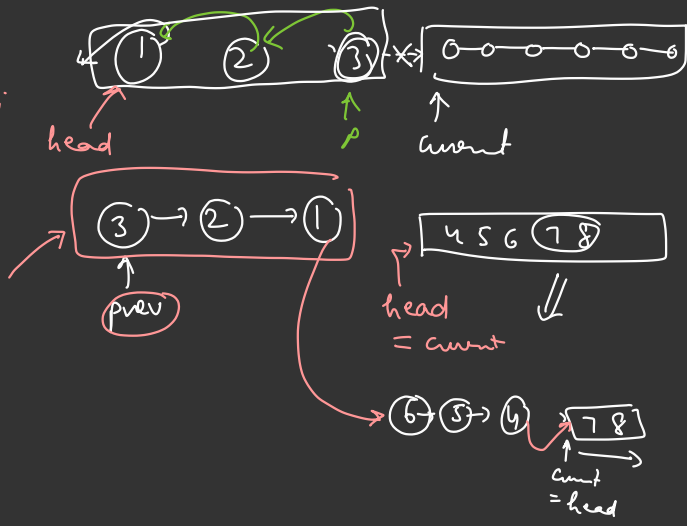
}

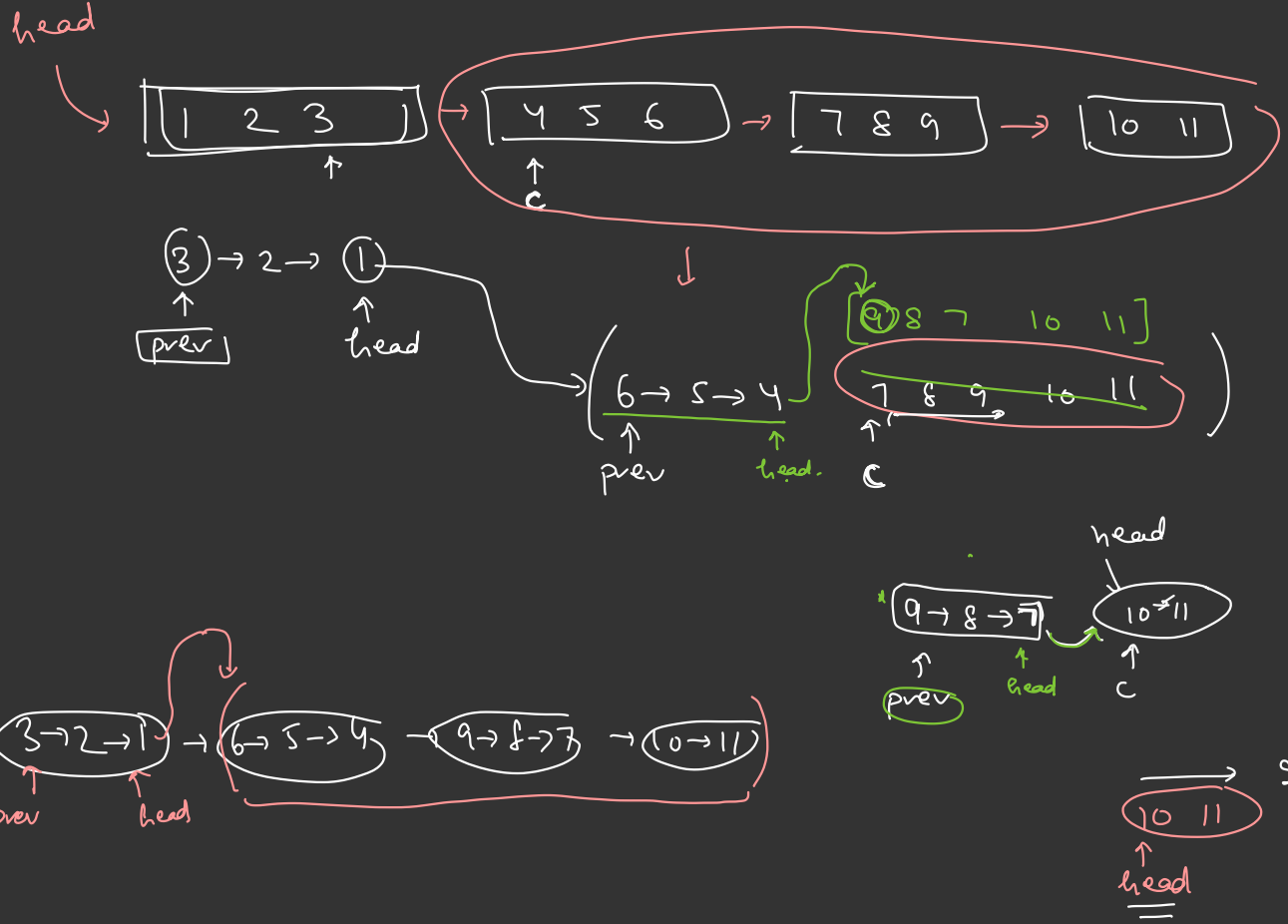
New head

{

return prev;

}




head
1 2 3
4 5 6
7 8 9
10 11
c
3 → 2 → 1
prev
head
6 → 5 → 4
prev
head
c
9 → 8 → 7
10 → 11
head
prev
head
c
head
3 → 2 → 1
6 → 5 → 4
9 → 8 → 7
10 → 11
prev
head
2 < 3 =
10 11
head