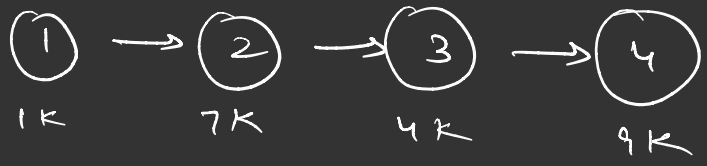
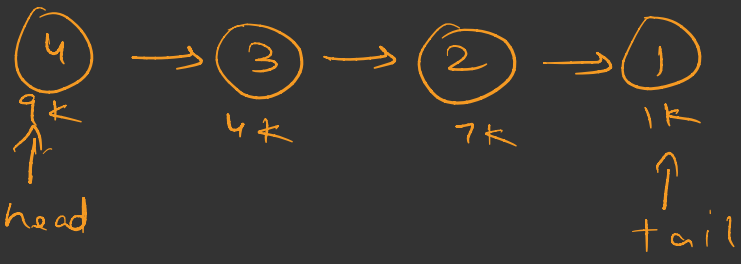
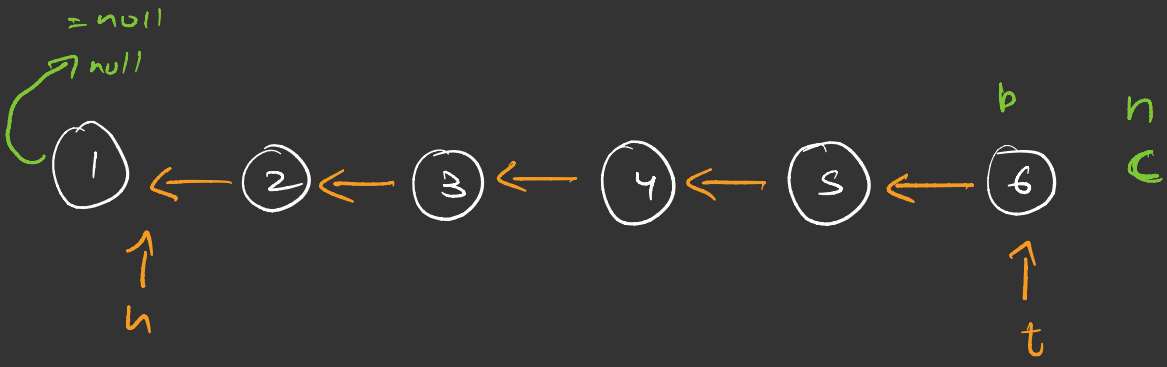
Ques Reverse Pointers Iteratively



⇒





p → null
c → head
n → c.n

$h = c.next$ (1) calculate next

$c.next = p$ (2) ask current to point to prev

$p = c$ (3) Update prev → curr

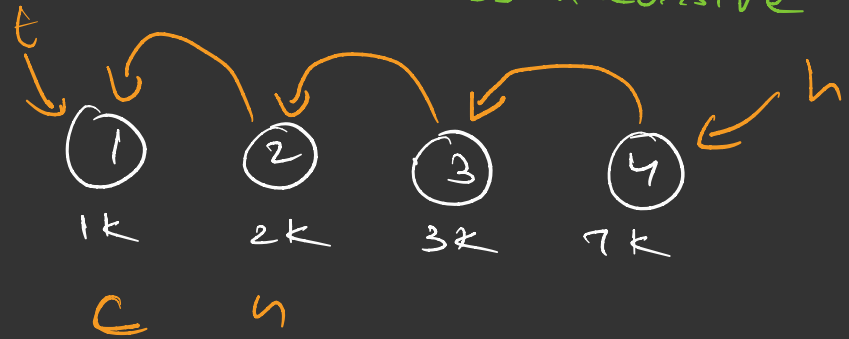
$c = n$ (4) Update curr → next

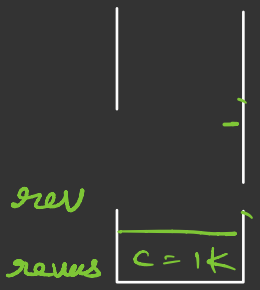
Node R = head

head = tail

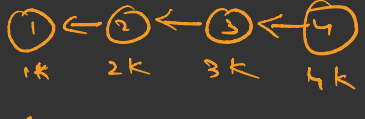
tail = R

Reverse Pointers Recursive





Reverse PR (Node c)



Exp.

Faith

reverse (c.n)

Combine

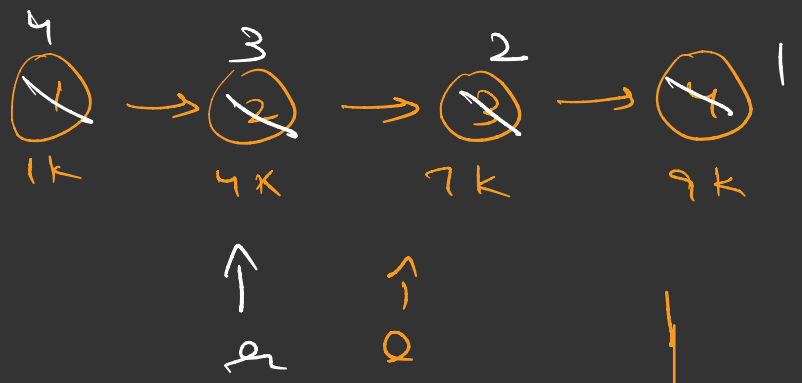
reverse (c.next);

Node n = c.next;

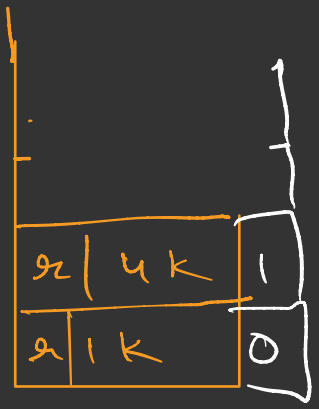
n.next = curr

c.next = null

Ques Reverse Data Recursively



Q if (i >= size/2)



Ques Mid of a Linked List

(1) size is given → go till size/2 Node
get ((size/2) - 1),

(2) slow Fast

```
slow = head
fast = head.next

while (fast != null && fast.next != null)
{
    slow = slow.next;
    fast = fast.next.next;
}
return slow.data
```

Ques Remove duplicates from sorted LL

temp.data == temp.next.data