Ques Mid of linked list

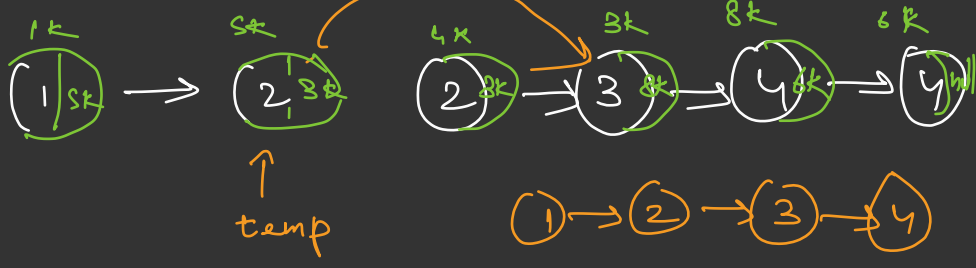
① Size is given → go till size/2
⇒ getNode((size/2) - 1)

② Slow Fast

slow → head
fast → head.next

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

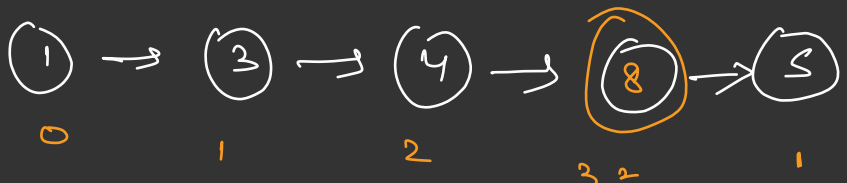
Ques Remove duplicates in Sorted LL



temp.next = temp.next.next

Ques kth node from end

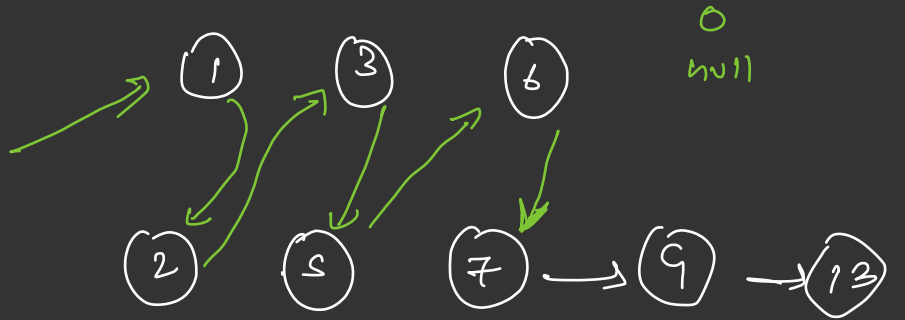
k = 2

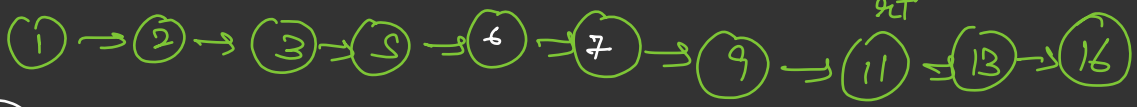


getNode (size - k)

S - 2
⇒ 3

Ques Merge 2 sorted LL





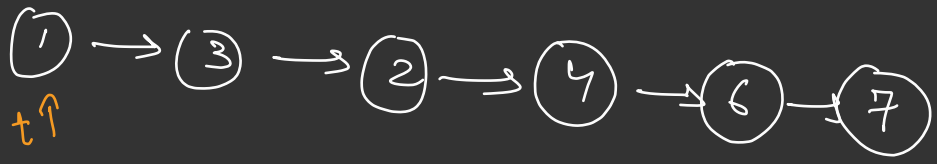
LL ⇒

comparing

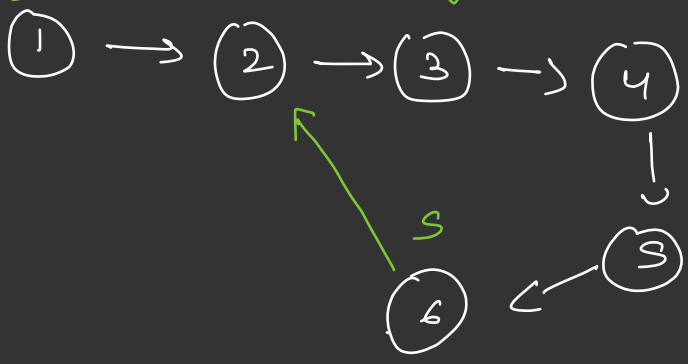
```
r.tail.next = t;
while(t != null)
{ r.tail = t
  r.size++;
  t = t.next;
}
```

```
if (o.data < t.data) {
    rt.next = o;
    o = o.next;
    rt = rt.next
} else {
    rt.next = t;
    t = t.next;
    rt = rt.next;
}
```

Ques Odd even



Ques Detect cycle/loop

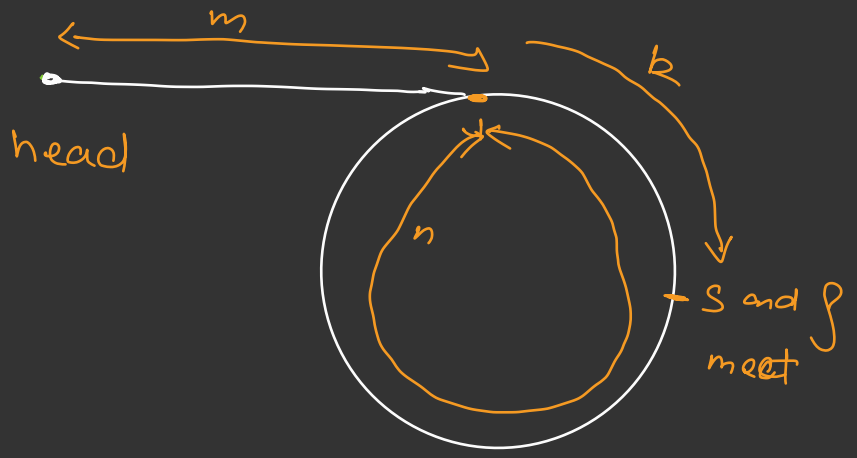


hasCycle

```
while(fast!=null
      &&fast.next!=)
{fast = fast.next.next
 slow = slow.next
 if (fast == slow)
```

Ques length of cycle

Ques starting point of cycle



dist travelled by J = 2x distance travelled by S

$$m + in + k = 2 \times (m + in + k)$$

$$m + in + k = 2m + 2jn + 2k$$

$$(i - 2j)n = m + k$$

$$\Rightarrow \underbrace{m+k}_{\text{factor of } n} = \underbrace{(i-2j)n}_{n}$$

$m + k$ = factor of n

$$\Rightarrow m = \text{multiple of } n - k$$

→ updateSize(n.next)

size = 6

```
LinkedList odd = new LinkedList();
LinkedList even = new LinkedList();

while(this.isEmpty() != true) {
    Node node = this.removeFirst();
    if(node.data % 2 == 0) { // add
        even.addLast(node);
    } else { // add in oddList
        odd.addLast(node);
    }
}

odd.tail.next = even.head;
this.head = odd.head;
this.tail = even.tail;
this.size = odd.size + even.size;
```