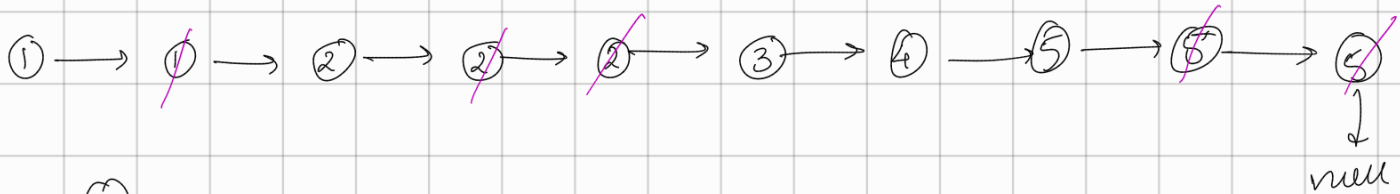
→ Remove duplicates - I & II

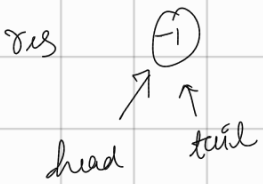
→ Fold LL , Unfold LL

→ Odd Even LL → Index
↳ Value

→ Rotate List.

① Remove duplicates { Sorted }





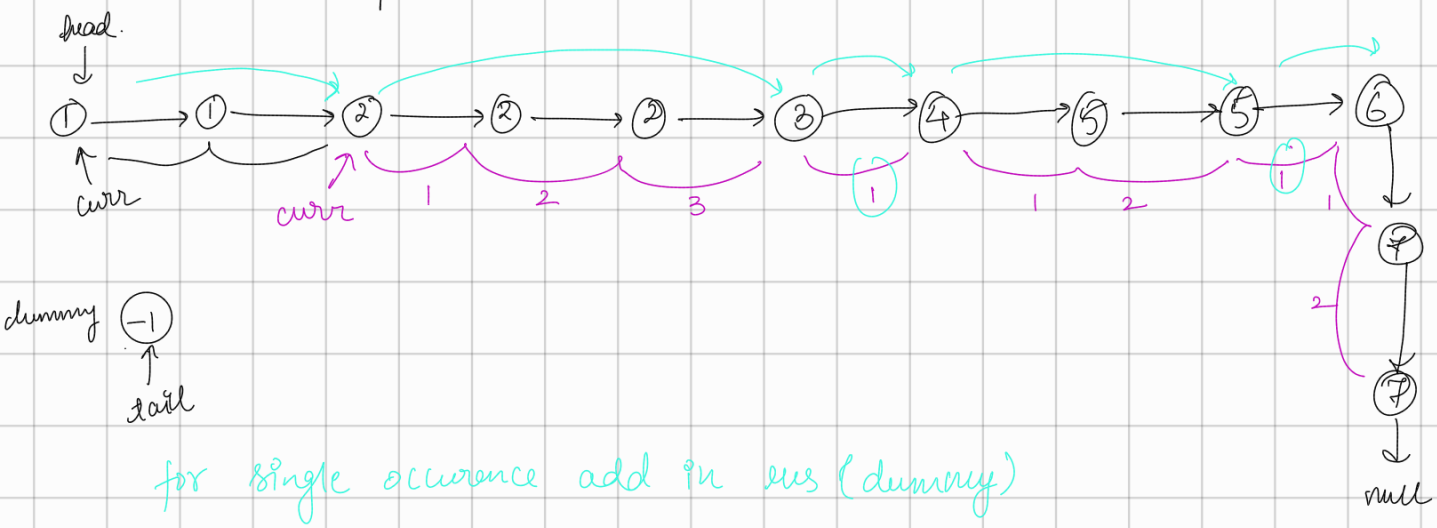
whenever new node is encountered,
do { tail.next = head
tail = tail.next }
then update head.next for all
consecutive nodes having same data

CODE:

```
ListNode dummy = new ListNode(-1), tail = dummy;
while (head != null) {
    if (tail == dummy || head.val != tail.val) {
        tail.next = head;
        tail = head;
    }
    head = head.next;
}
return dummy.next;
```

TC: O(N) SC: O(1)

(3) Remove all duplicates



for single occurence add in res (dummy)

Update curr = curr.next till curr.val != head.val

(6) Fold LL {Reorder LL 143}

Approach: find middle of list, detach from there reverse the right half & merge both again

(4) Unfold LL

Separate alternate lists, reverse the second one & attach it to first list's tail

(5) Odd Even LL by Value

Approach: make 2 different lists & add them at last.

Eg: E → O → E → E → O → O → null

res: E → E → E ⇒ O → O → O → null

CODE:

```
Node dummyEven = new Node(-1), dummyOdd = new Node(-1);
Node evenTail = dummyEven, oddTail = dummyOdd;
while(N-- > 0) {
```

```
if (head.data % 2 == 0) {
    evenTail.next = head;
    evenTail = head;
} else {
    oddTail.next = head;
    oddTail = head;
}
head = head.next;
}
evenTail.next = dummyOdd.next;
oddTail.next = null;
return dummyEven.next;
```

⑥ Odd Even LL by Index

Same as above instead check for index.

CODE:

```
Node dummyEven = new Node(-1), dummyOdd = new Node(-1);
Node evenTail = dummyEven, oddTail = dummyOdd;
int count = 0;
while(N-- > 0) {
    count++
if (count % 2 == 0){
    evenTail.next = head;
    evenTail = head;
} else {
    oddTail.next = head;
    oddTail = head;
}
head = head.next;
}
```

```
evenTail.next = dummyOdd.next;
OddTail.next = null;
return dummyEven.next;
```

⑦ Rotate LL

Brute Force: for each count ∈ [1, k]
pick the last Node & place it at beginning

TC → O(N × k) ∵ k Linked List traversals

Can be optimised using DLL to O(N + k)

Optimized: find size of array (n)
do k = k % n to reduce redundant rotations.
and only rotate once by
→ pick the last k nodes group from the given linked list
→ add this group as is to the beginning of the list
→ update pointers accordingly
→ return the new head as head of rotated list.

TC: O(N), SC = O(1)
↳ Not O(N + K)
∵ k = k % N ⇒ k < N