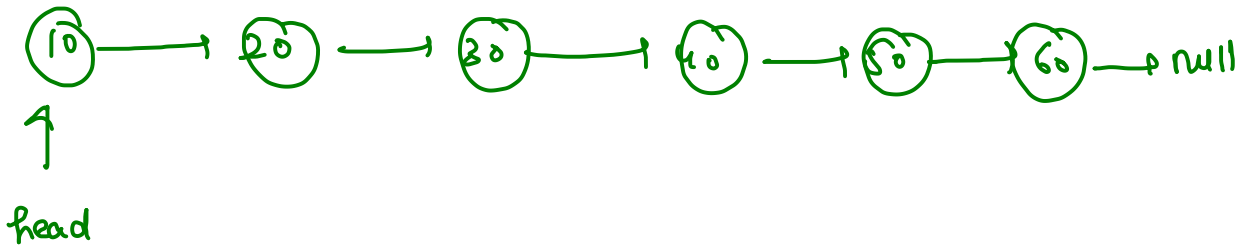
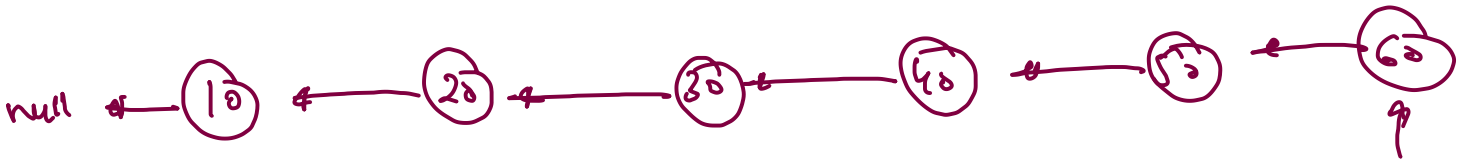
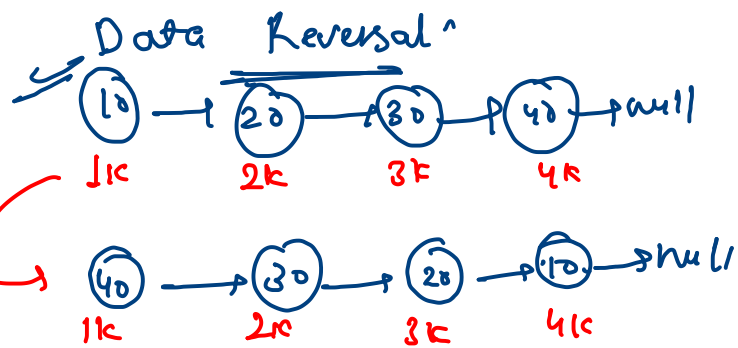
Reverse of a LinkedList →

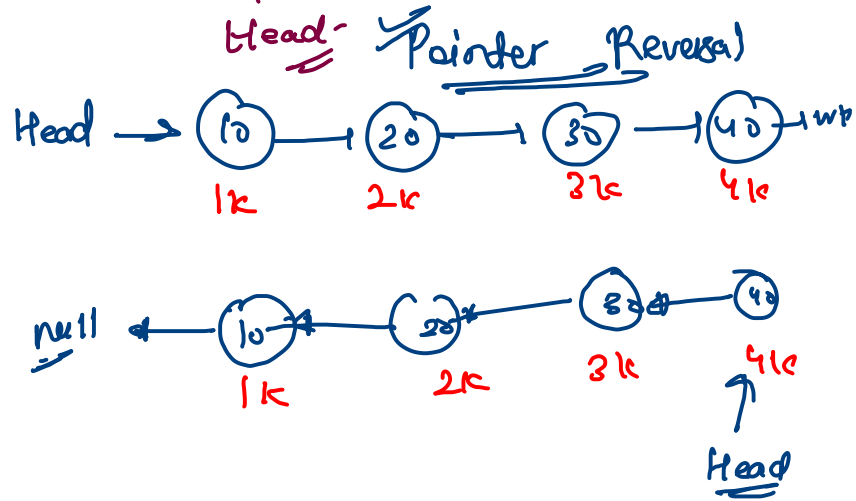




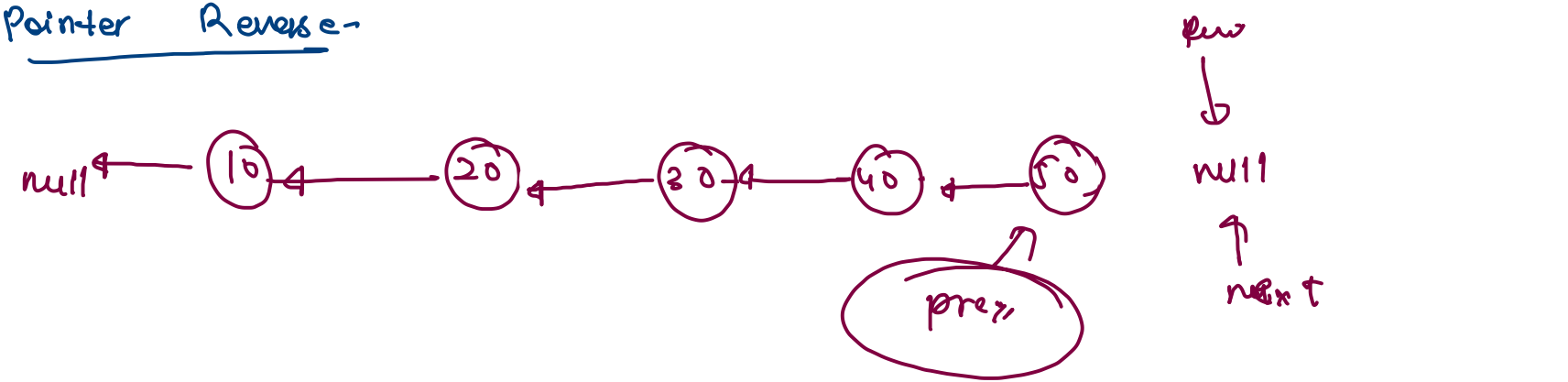
(1) Data →

(2) Pointer.





Pointer Reverse -



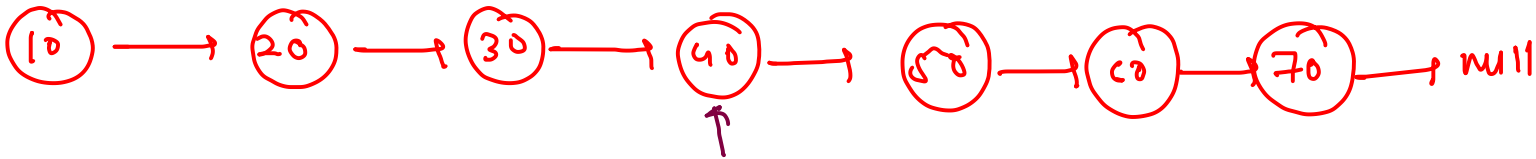

```
while(curr != null) {
    Next = curr.next;
    curr.next = prev;
    prev = curr;
    curr = next
}
head = prev;
```

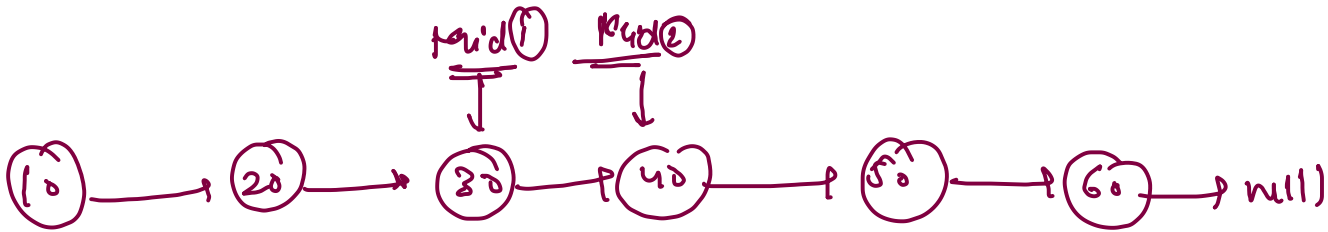
curr = null - Null ptr to a new connection

validation

shifting

Middle of linked list





speed = $\frac{dist}{time}$

dist = speed x time.

slow = x

time = t

distance = xt

fast 2x

time = t

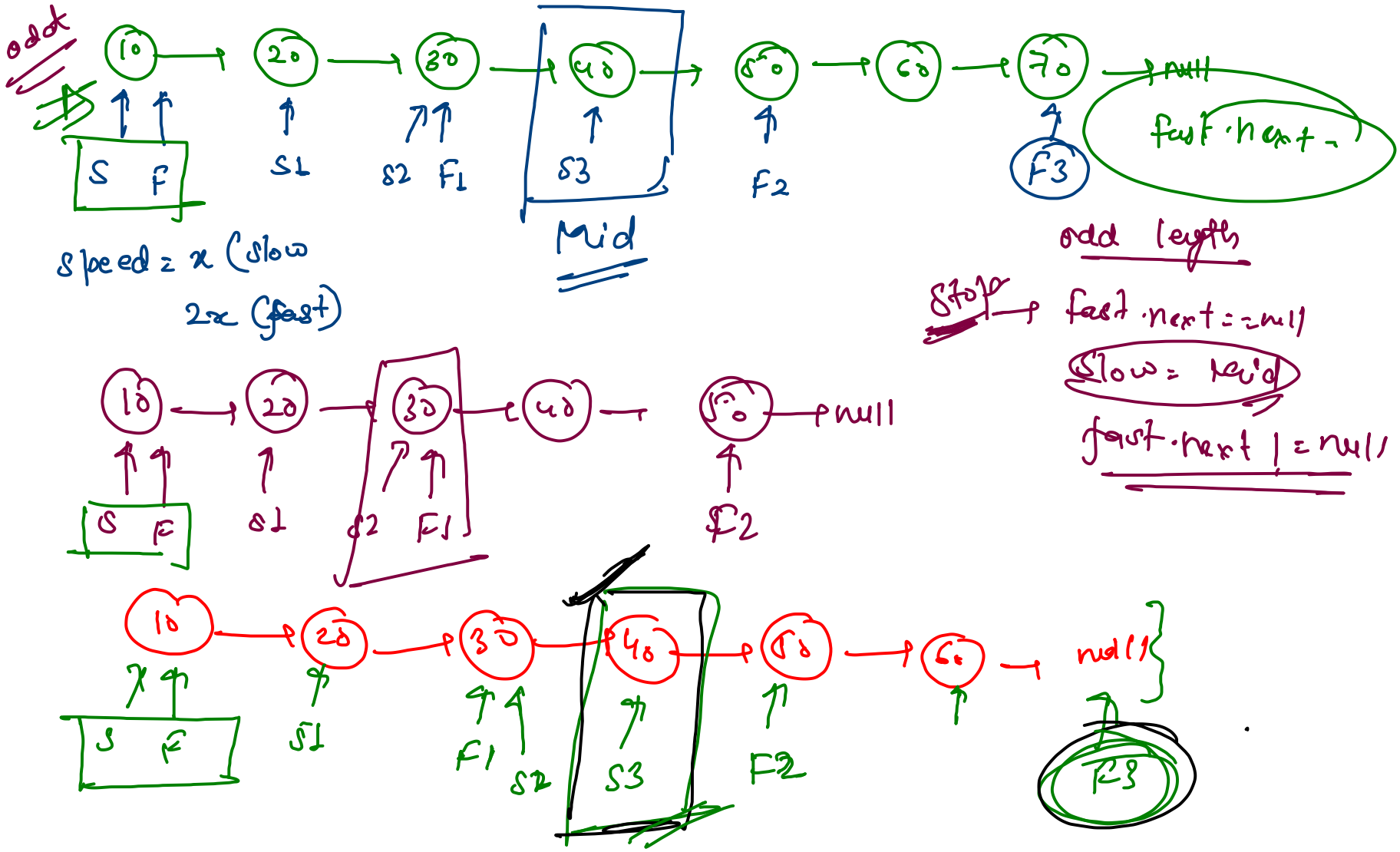
distance = 2x x t

= 2xt

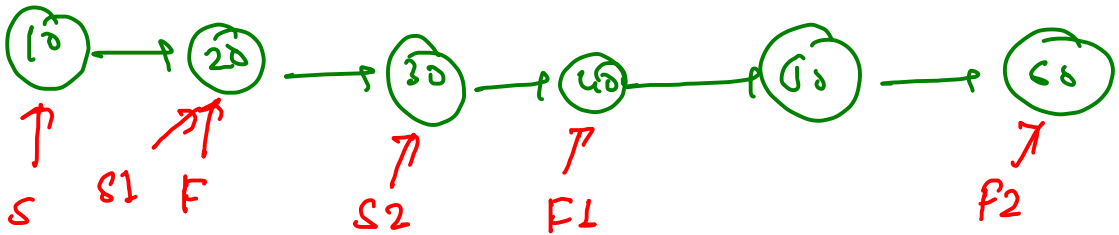
slow → mid

fast → end

2 * x t = 2xt


odd
10
20
30
40
50
60
70
null
S
F
S1
S2
F1
S3
Mid
F2
F3
fast.next.
Speed = x (slow
2x (fast)
odd length
Stop
fast.next==null
slow = mid
fast.next != null
10
20
30
40
50
null
S
F
S1
S2
F1
F2
10
20
30
40
50
60
null
S
F
S1
F1
S2
S3
F2
F3

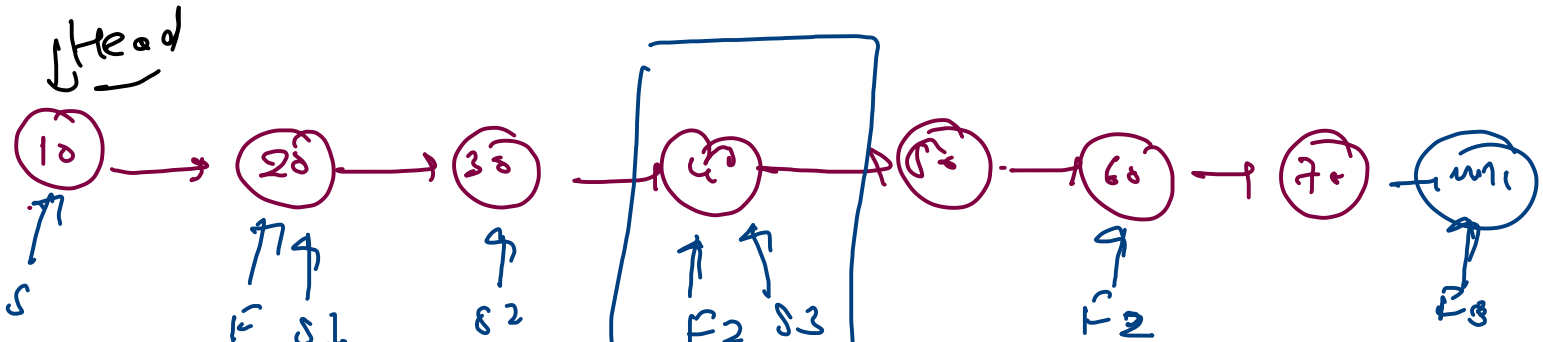
Even



→ F2.next == null

S2 → mid

odd



Initial

slow = Head;

fast = Head.next;

(fast != null && fast.next != null) {

slow = slow.next

fast = fast.next.next;

}

→ fast == null

slow → mid } odd

} even

Is palindrome

Space - O(n)

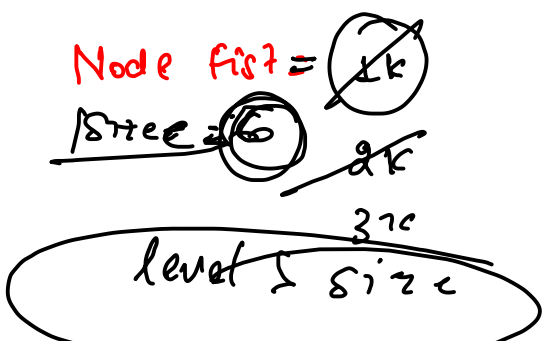
time - O(n)

1. Array - fill

   check in

   Array.

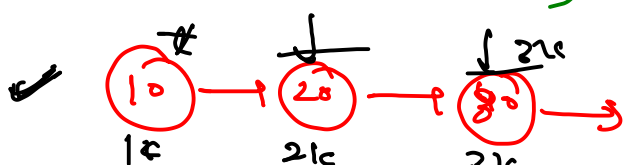
2. Recursion.

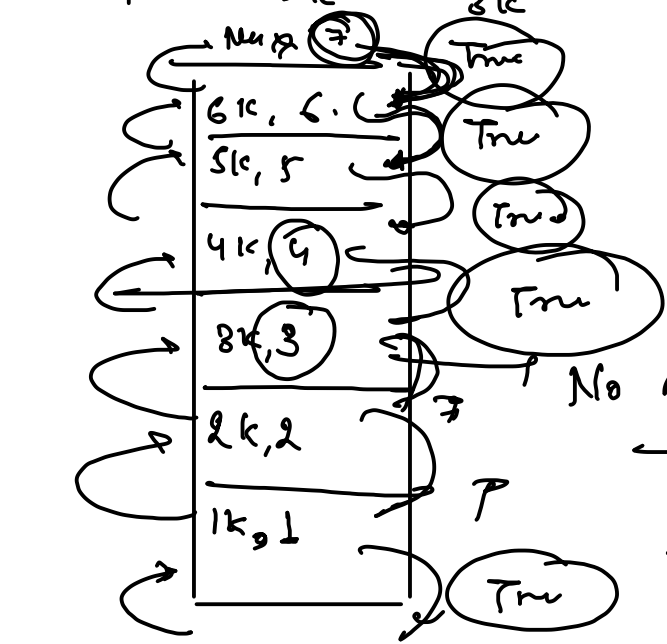


Space - O(1)

time - O(n²)

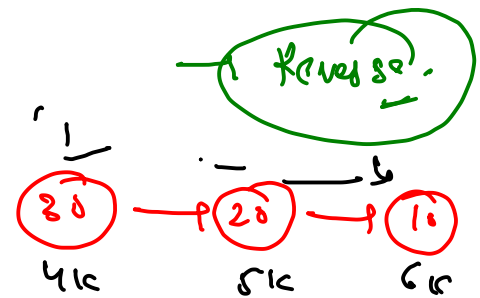
Get Node At( )





Space (o|1)

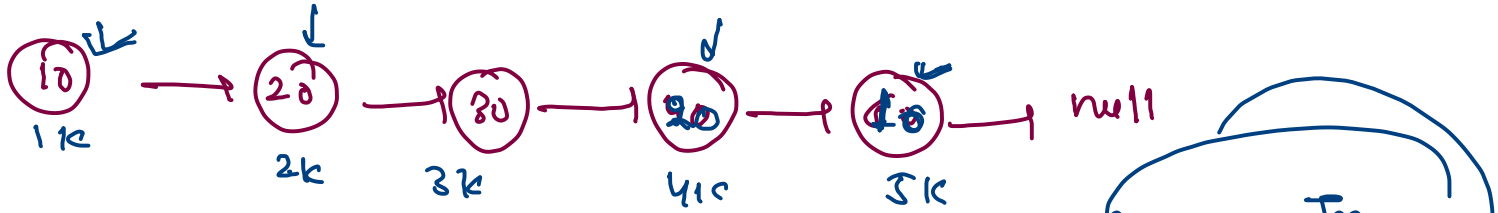
time - O(n)



post = check if equal.

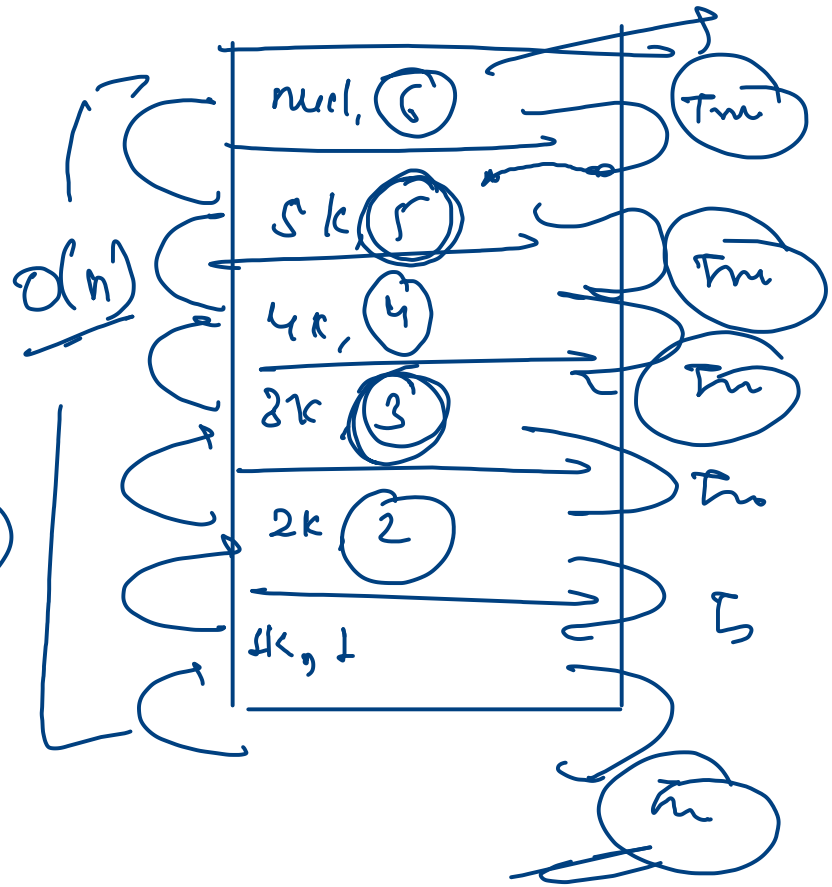
fist = first next

No need of check



Base → True

recur

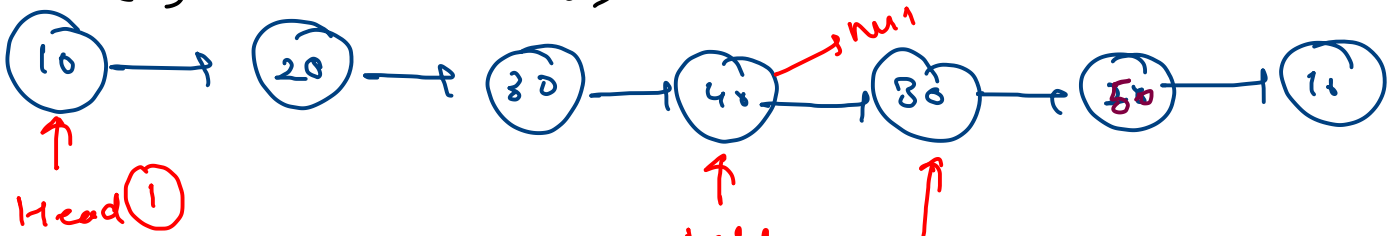


left = ~~1k~~ 2k 3k

size = 5

5/2 = 2

Time → O(n) Space - O(1)



Even : first Middle

① find Mid. n/2

② Split from mid.nex

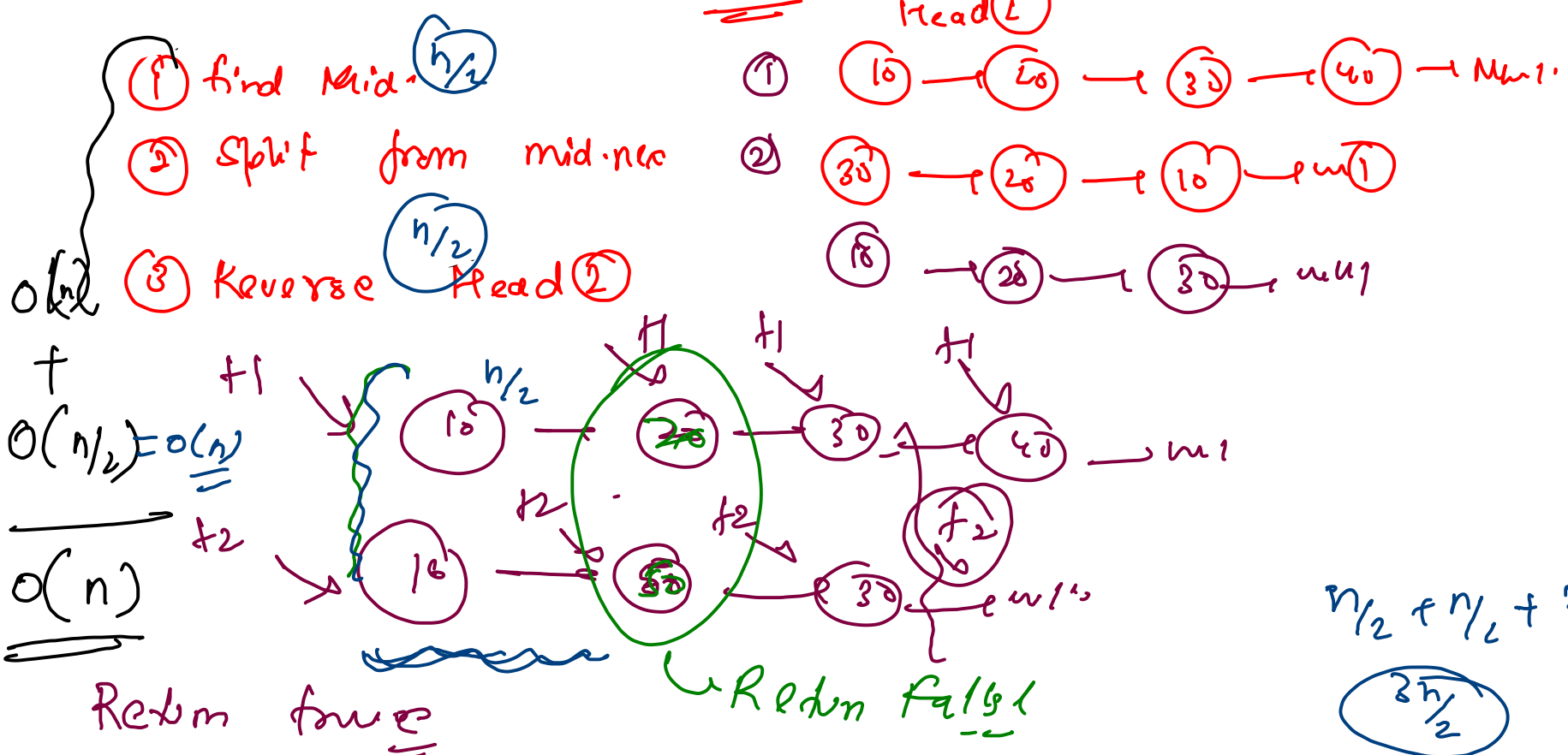
③ Reverse Head (2) n/2

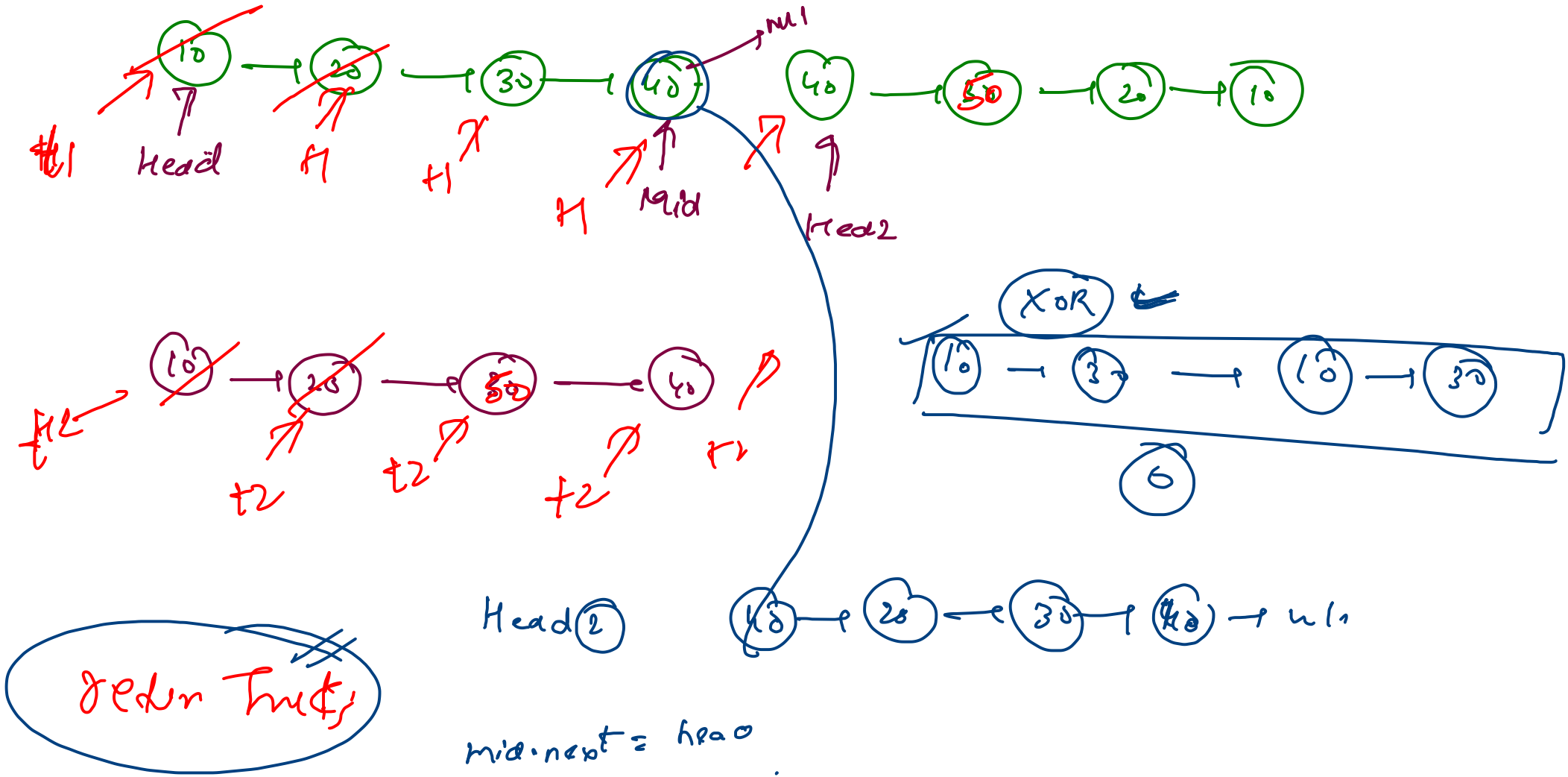
O(n)

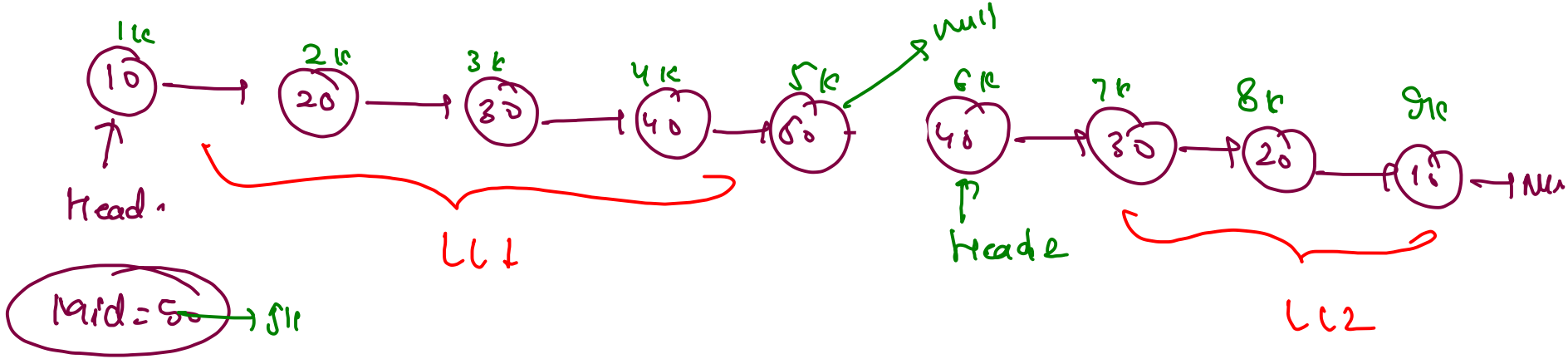
+

O(n/2) = O(n)

O(n)



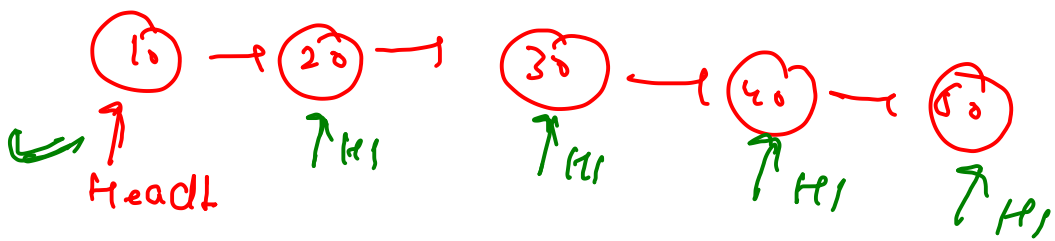

10
20
30
40
null
t1
Head
t1
t1
t1
Mid
Head2
40
50
20
10
10
20
50
40
t2
t2
t2
t2
t2
XOR
10
3
10
30
6
Head 2
40
20
30
40
null
Dedun Tricks
mid.next = head

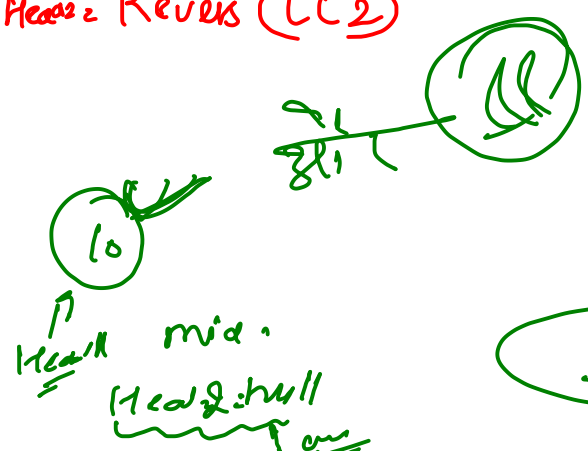


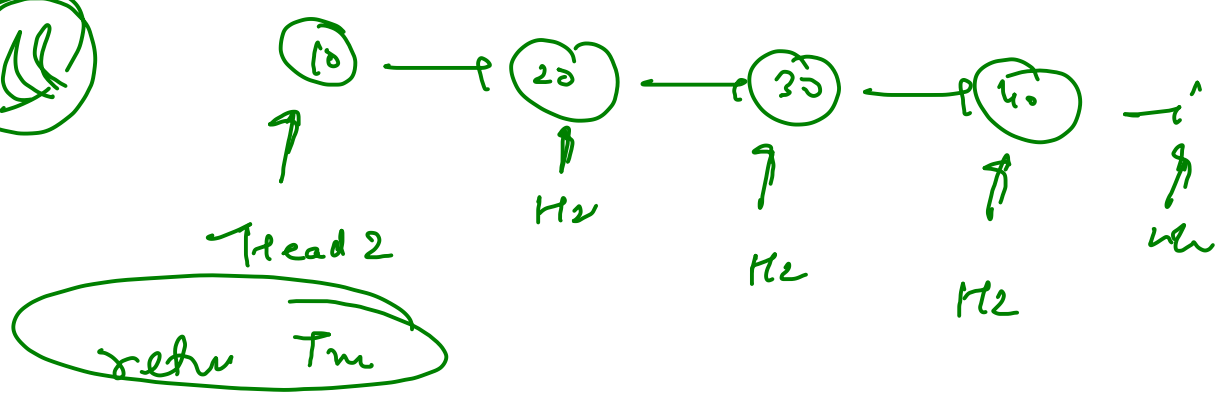
Head2 = mid.next

mid.next = null

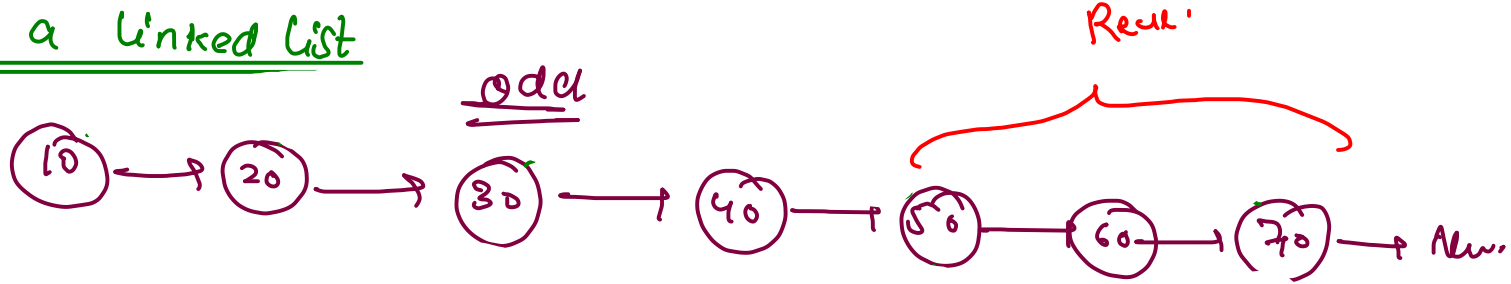
Head2 = Reverse (LL2)

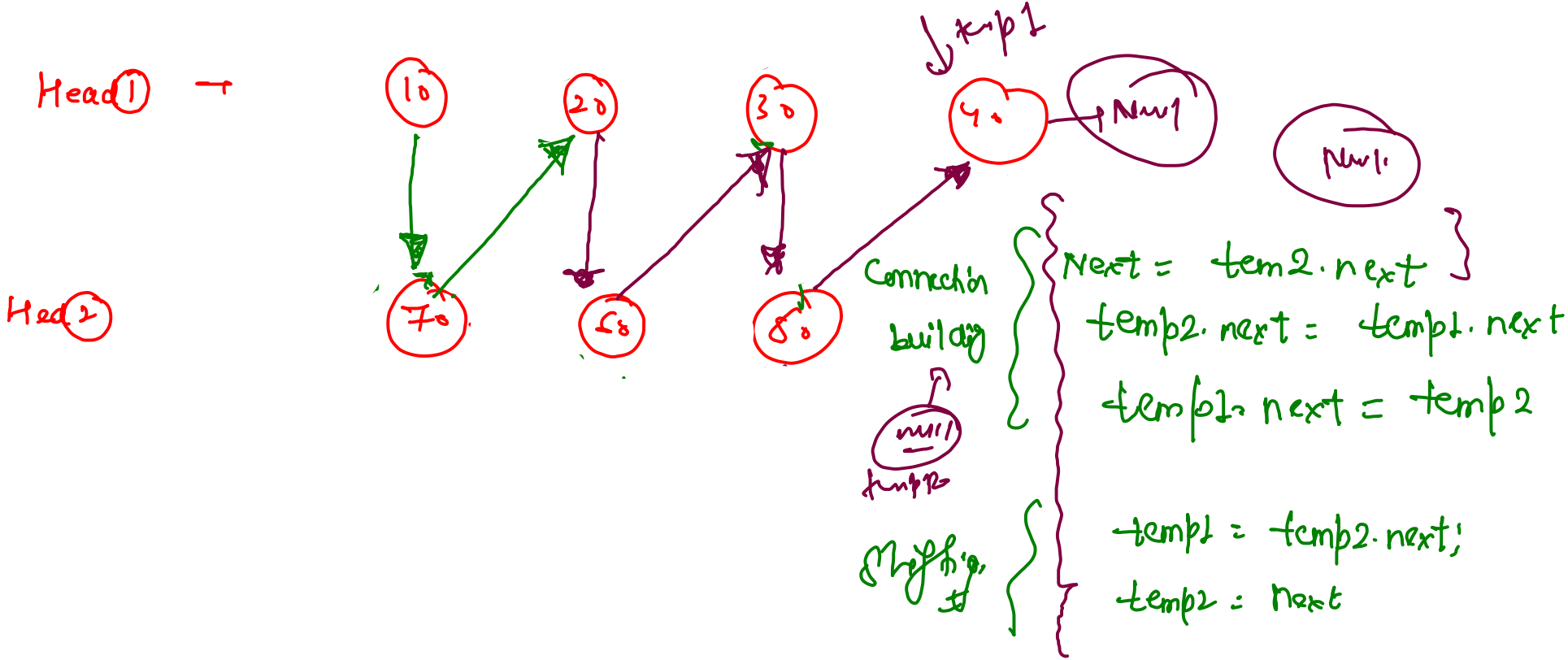






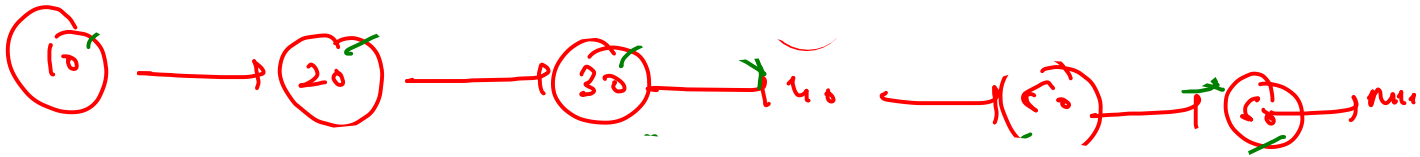
Fold a Linked List

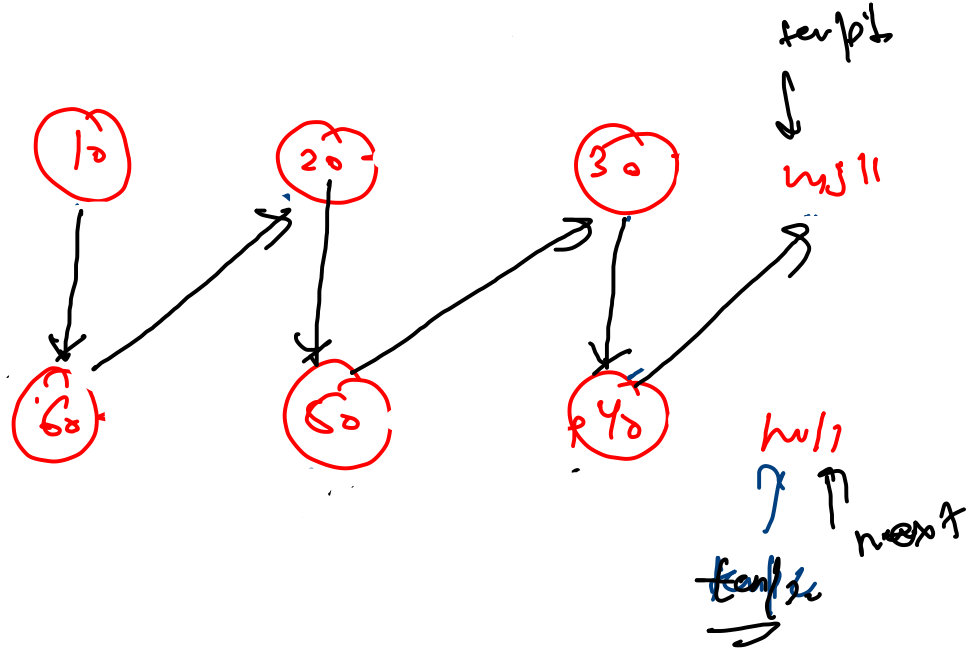




Even Size.

Steps ① get Mid.
② Make head2 & unlink mid with h2.
③ Revers h2.
④ → fold logic





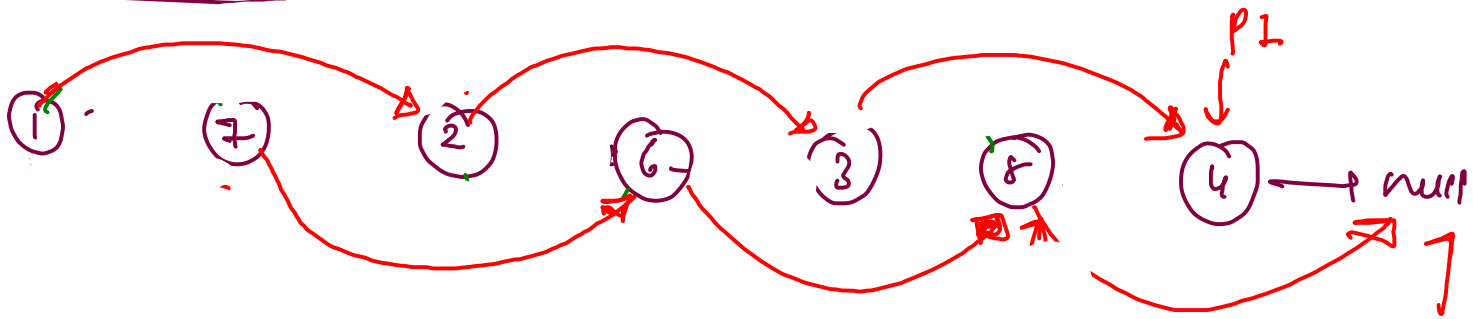
while (temp1 != null && temp2

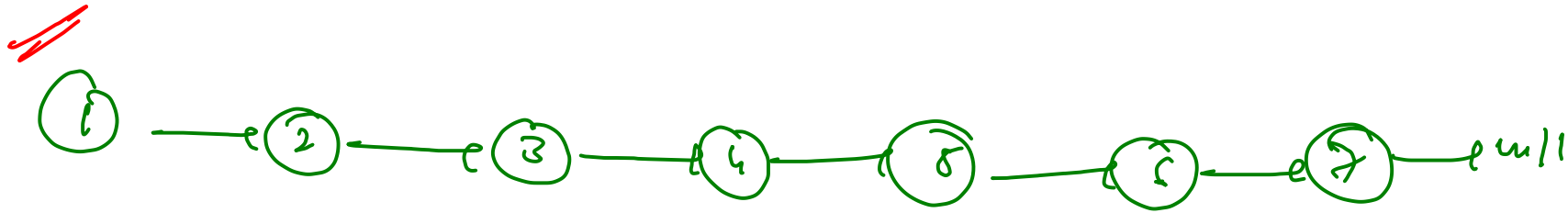
```
Next = temp2.next
temp2.next = temp1.next
temp1.next = temp2

temp1 = temp2.next;
temp2 = next
```

Shifting

Unfold Linked List





① null

② Head

③ ① → ② 2 Nodes

④ odd

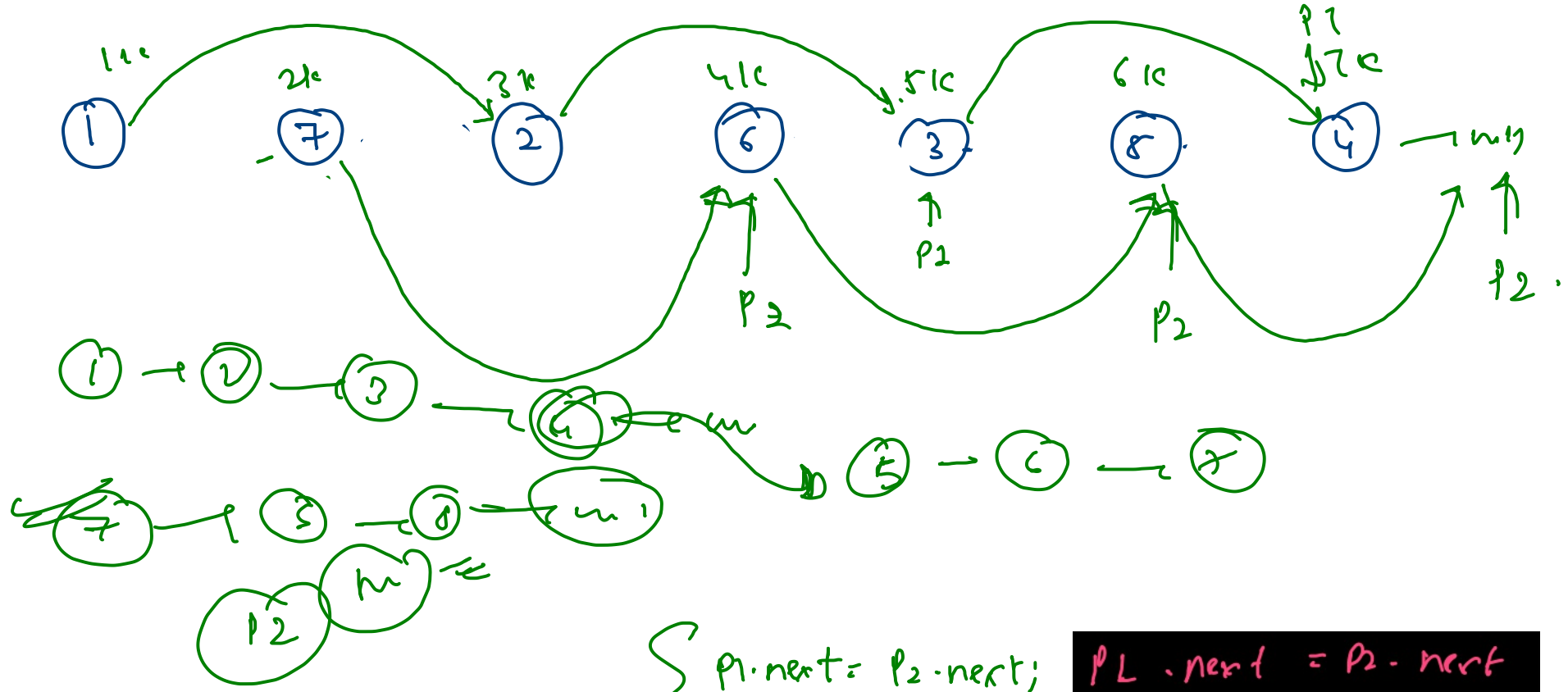
⑤ Even

```
pL.next = p2.next
p1 = p2.next;

p2.next = p1.next;
p2. = pL.next
```

```
pL.next != null &&

p2.next != null
```

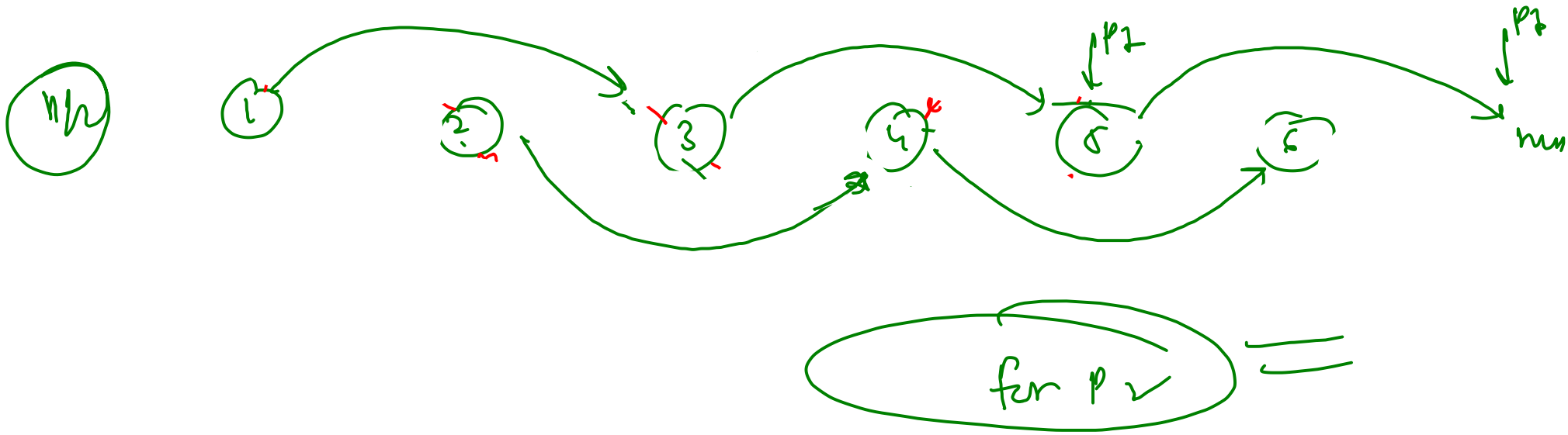


Head1 → 1c

Head2 → 2c

```
p1.next = p2.next;
p1 = p2.next;
p2.next = p1.next
p2 = p2.next;
```

```
p1.next = p2.next
p1 = p2.next;
p2.next = p1.next;
p2 = p1.next
```



1 — 2 — 3 — 4 — 5 — 1

1 — 3 — 5 — 4 — 2