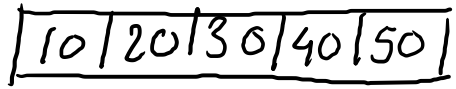
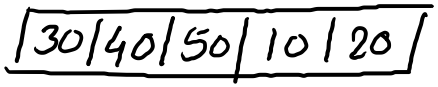
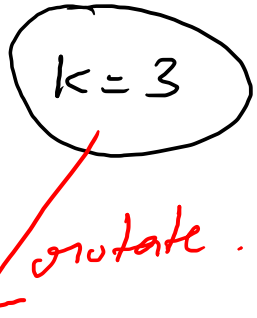
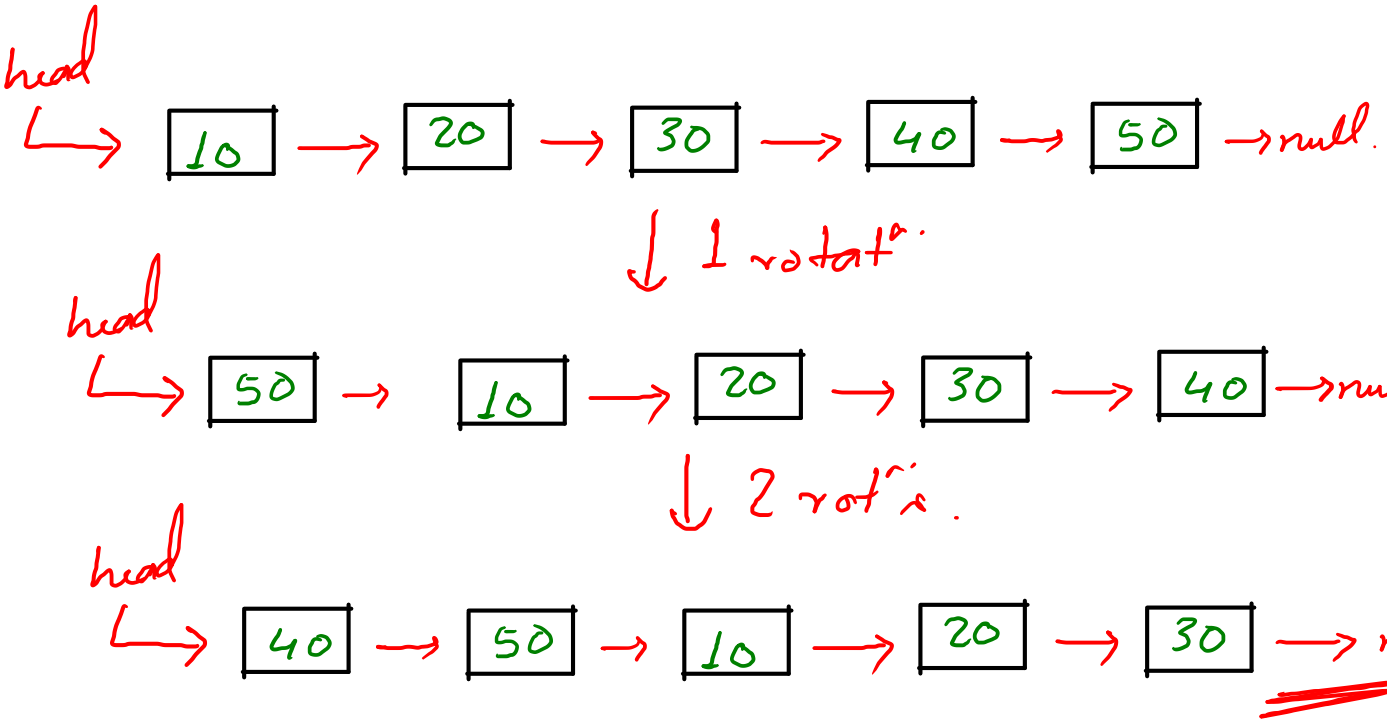
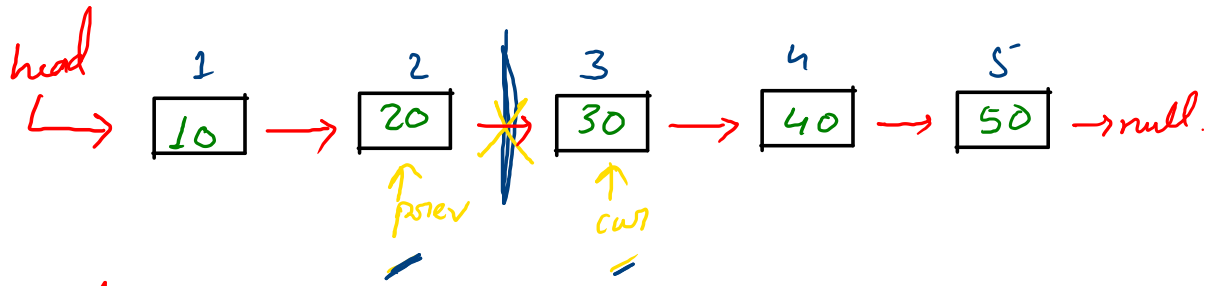
Rotate LL :-


10 20 30 40 50


30 40 50 10 20


k = 3
rotate.

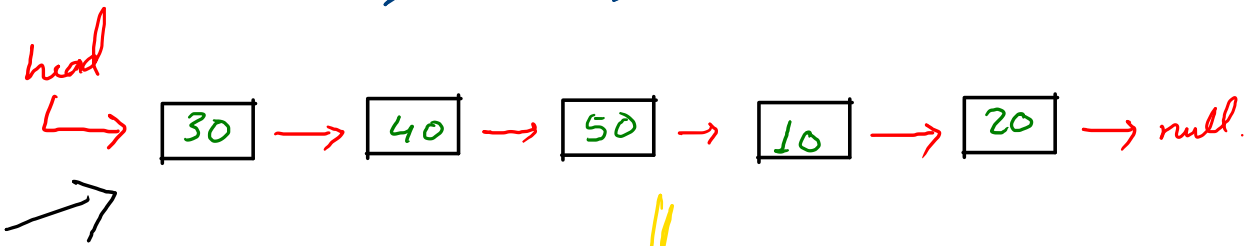

head
10
20
30
40
50
null.
1 rotate.
head
50
10
20
30
40
null.
2 rotate.
head
40
50
10
20
30
null.

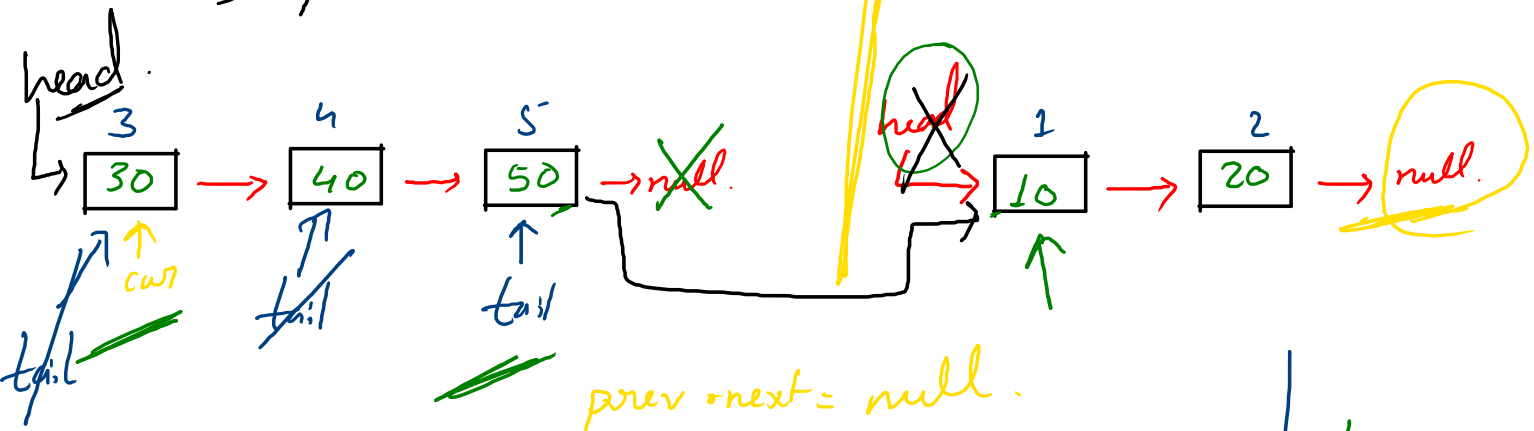


k = 3

n = 5

position = 5 - 3 = 2





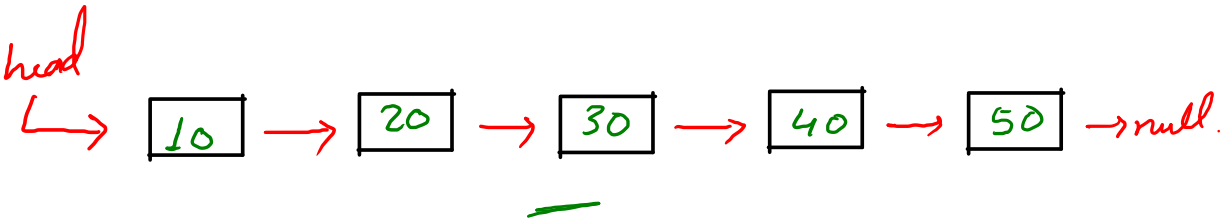
prev.next = null.

```
tail = curr
while (tail.next != null)
{   tail = tail.next
}
```

tail.next = head

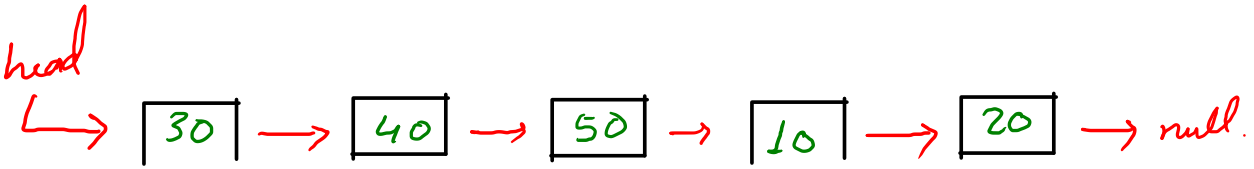
head = curr.

Pseudo code:-

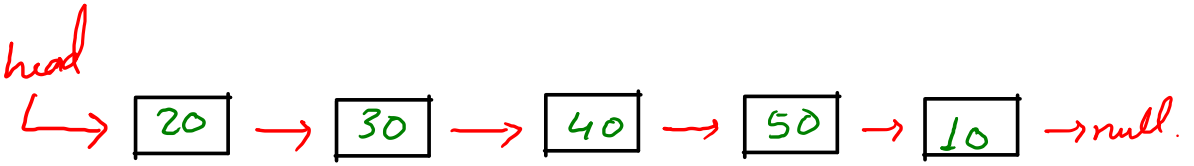


① find the length of LL as 'n'.

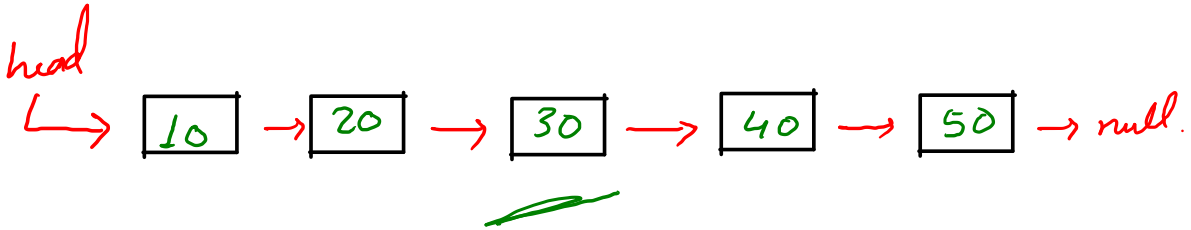
② if k=n, How many rotations!



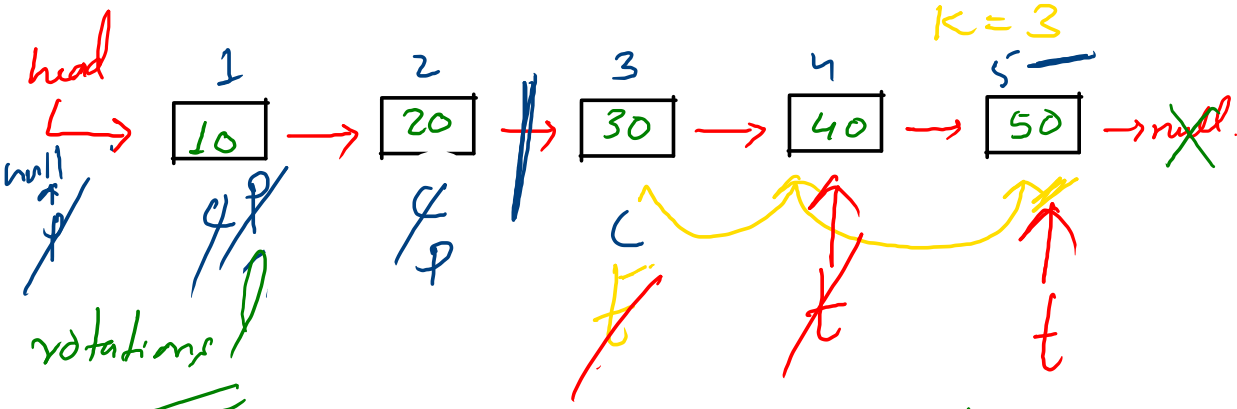
4 rotations.



5 times rotate.



Pseudo code:-



1. find the length of LL as `n`.
2. if k = n, How many rotations?

   $rot^n = k \% n$

3. position = $n - rot^n$.

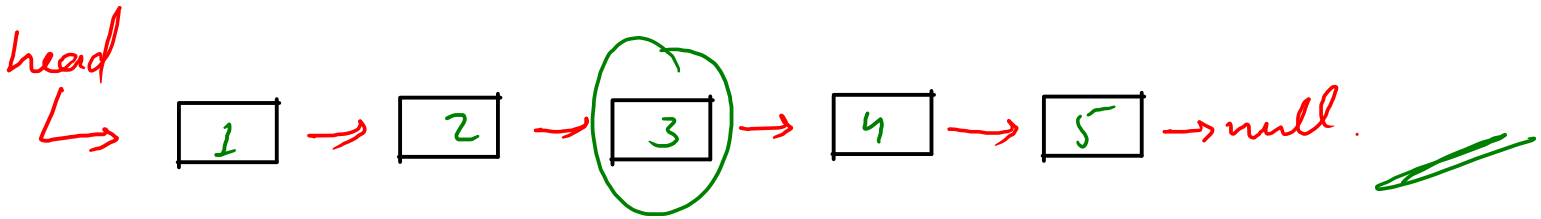
   position = 5 − 3 = 2

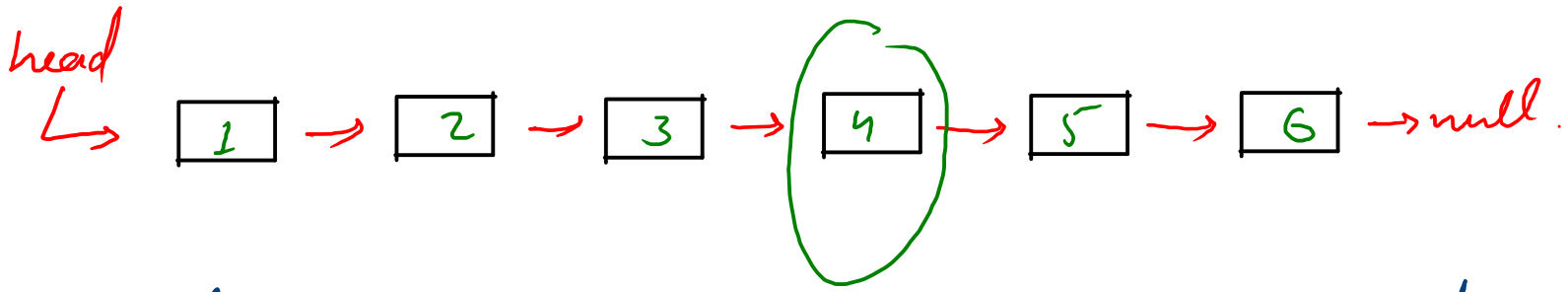
4. curr = head
   prev = null
   i = 0
   while(i < position)
   { prev = curr
   curr = curr.next, i++
   }

5. prev.next = null
   tail = curr
   while(tail.next != null)
   { tail = tail.next
   }
   tail.next = head
   head = curr.
   return head.

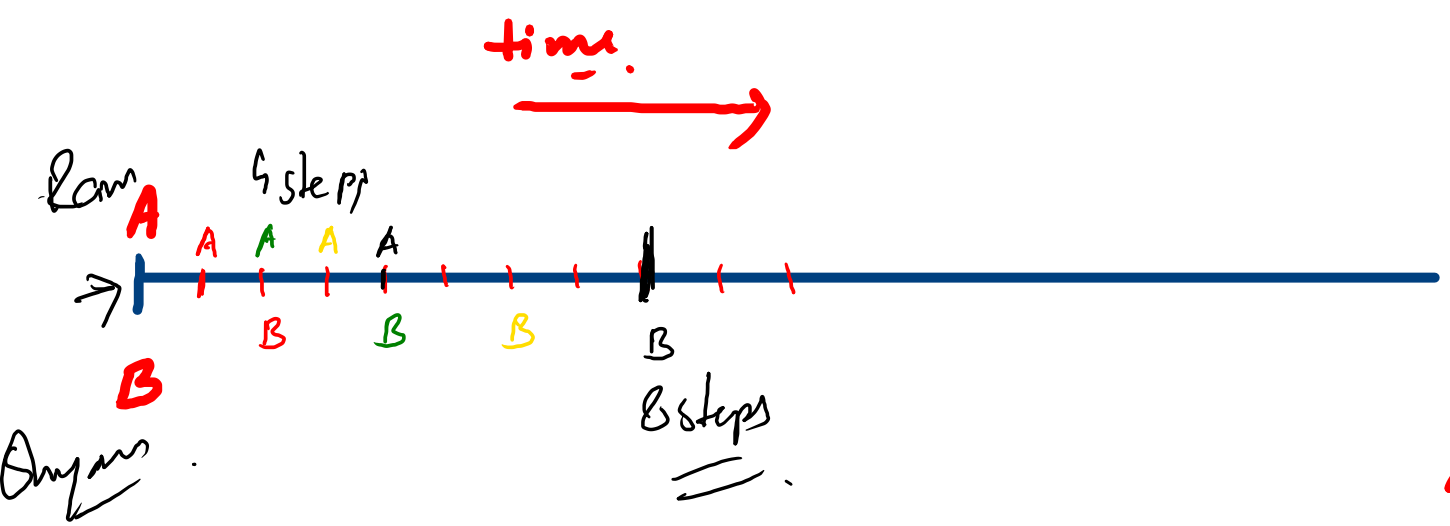
O(n)

Find Mid of LL:-





→ if there are two mid's return the 2nd one.



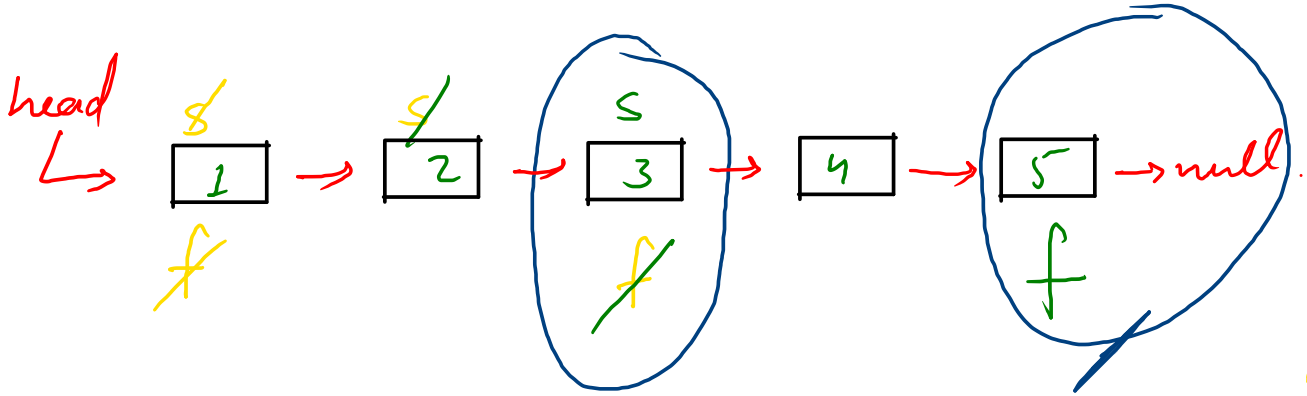
after 4 seconds

distance by A = 4 steps

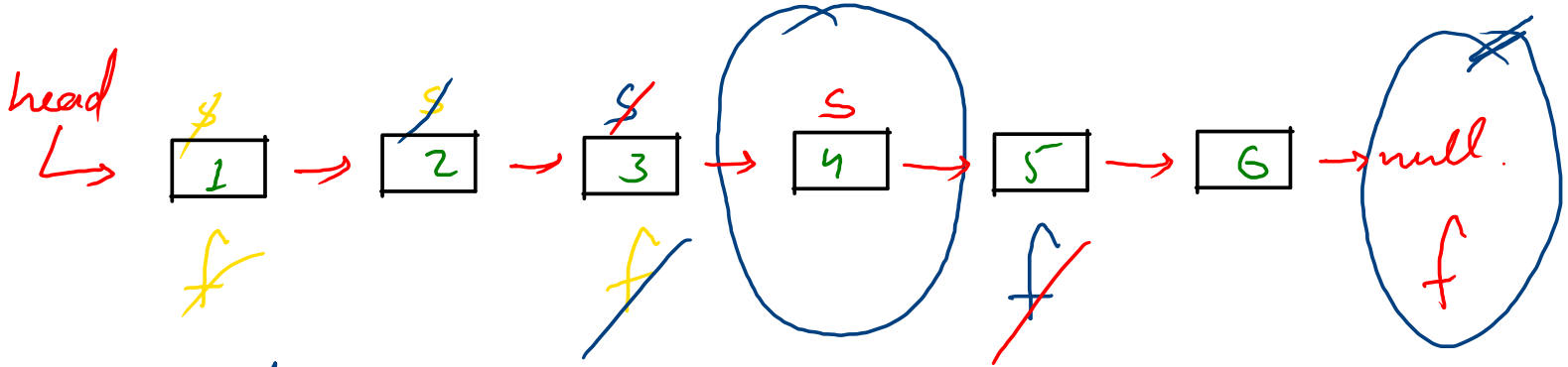
distance by B = 8 steps

A → 1 step at a time/sec.

B → 2 steps at a time/se.




```
while(fast!=null && fast.next!=null)
{ s=s.next
  f=f.next.next
}
return s
```

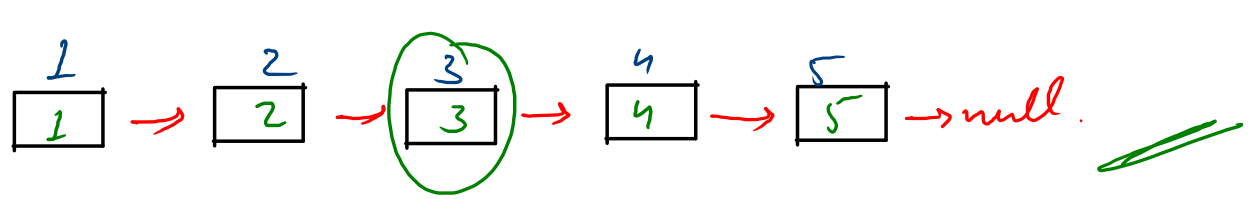


Hare & Turtle approach.

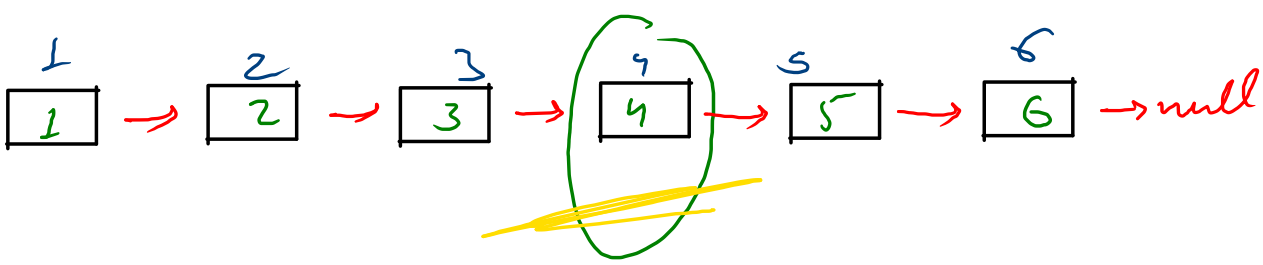
Pseudo code:

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

return slow
```

LL1



LL2



for LL1 → n = 5 → mid = floor(n/2) + 1
= ⌊5/2⌋ + 1 = 2 + 1 = 3

for LL2 → n = 6 → = ⌊6/2⌋ + 1 = 3 + 1 = 4

~~mid = n/2 = 6/2 = 6~~

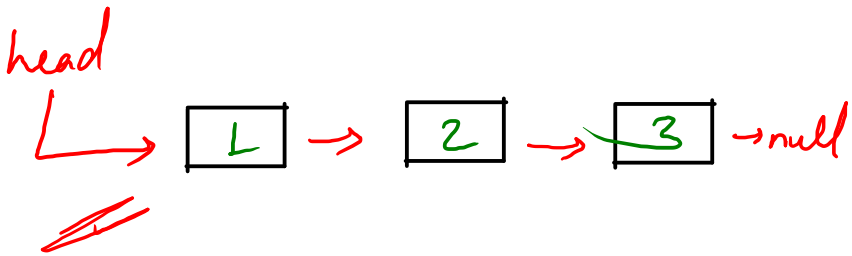
mid = ⌈n/2⌉ = ⌈5/2⌉ = 3

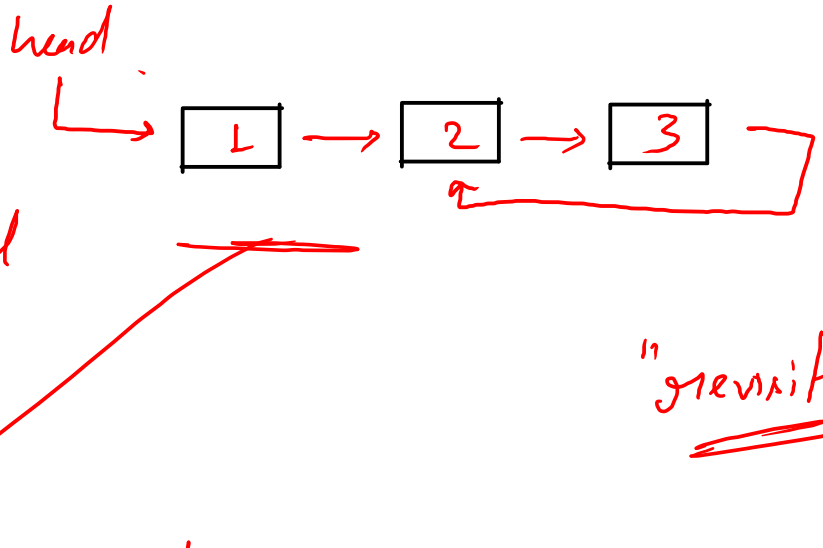
⌈6/2⌉ = 3

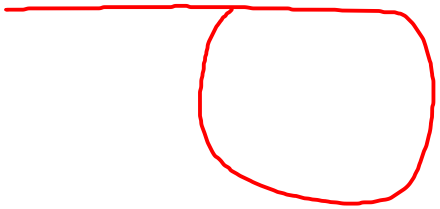
Steps

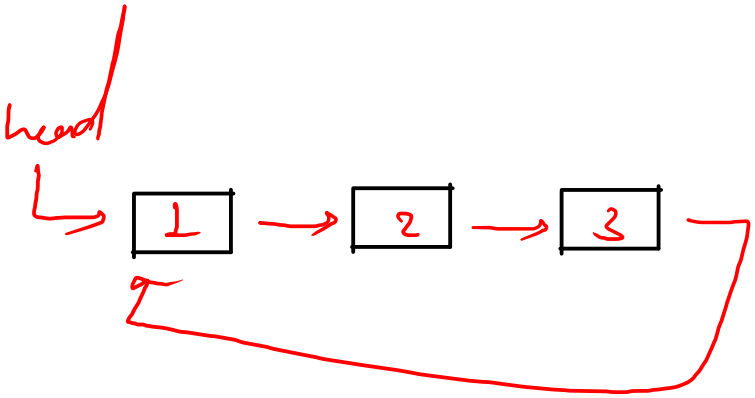
1. find the length as n
2. mid = floor(n/2) + 1
3. traverse till mid & return mid

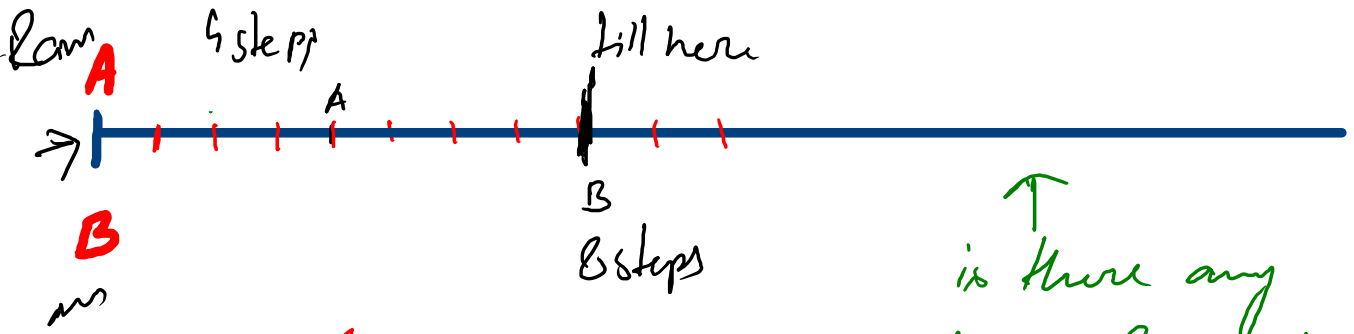
Detect Cycle in LL :-











A → 1 step at a time/sec.

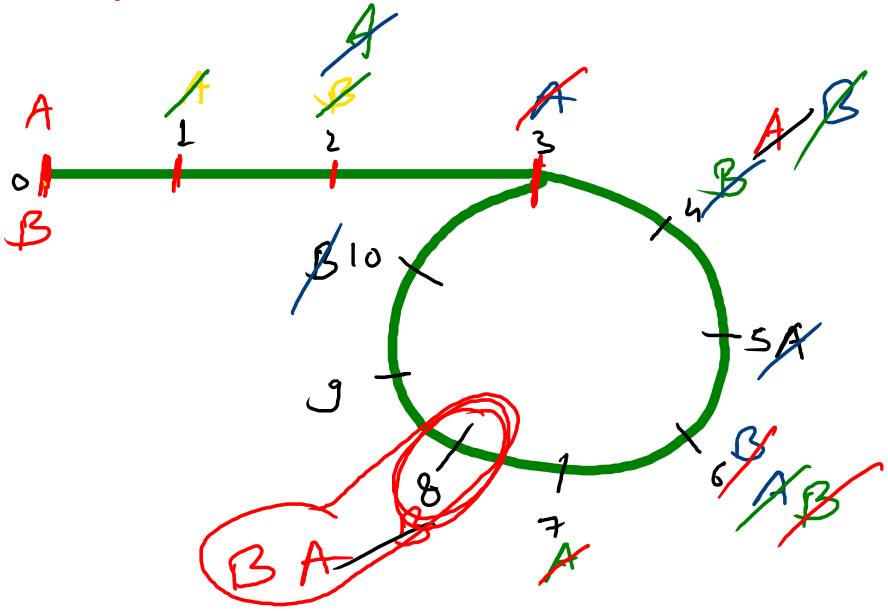
B → 2 steps at a time/sec.

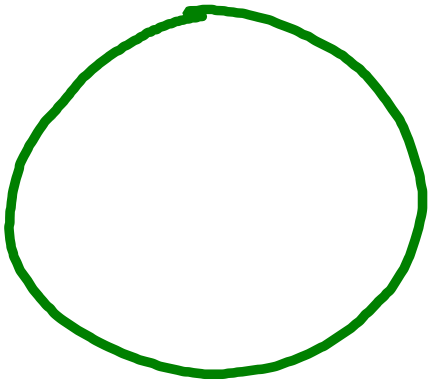
they both move simultaneously.

is there any chance B meets A!

never!

$O(n^2)$

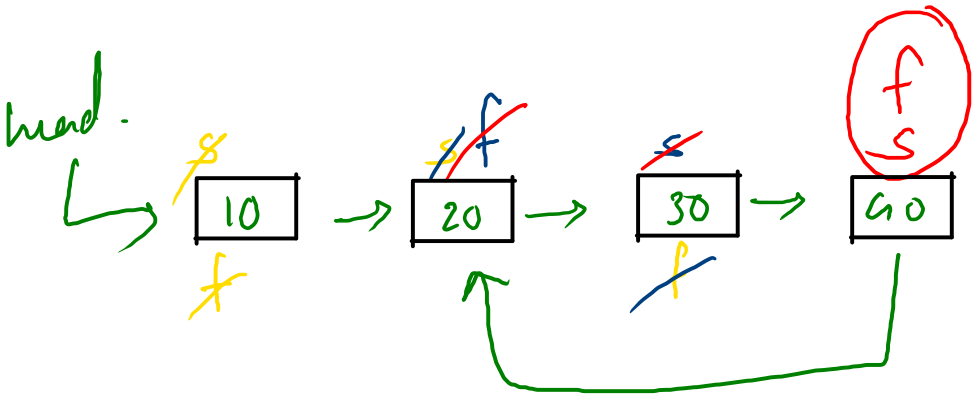




Floyd's Cycle Det. Algo

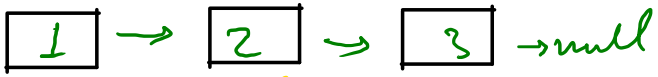
Pseudo Code :-

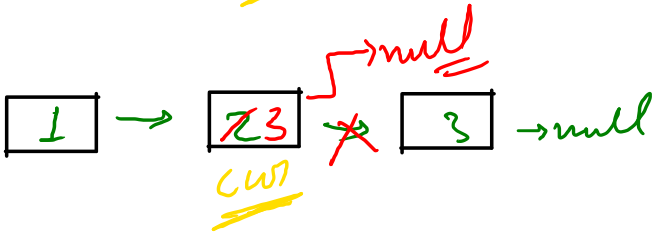
```
slow = head
fast = head

while (fast != null &&
       fast.next != null)
{
    slow = slow.next
    fast = fast.next.next
    if (fast == slow)
    {
        return true.
    }
}
return false.
```



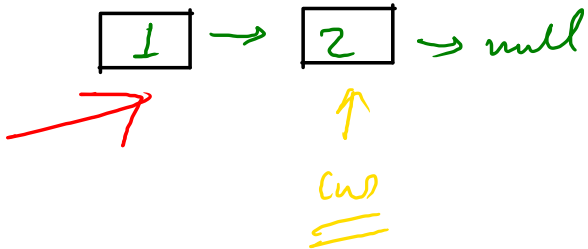
Delete node without head :

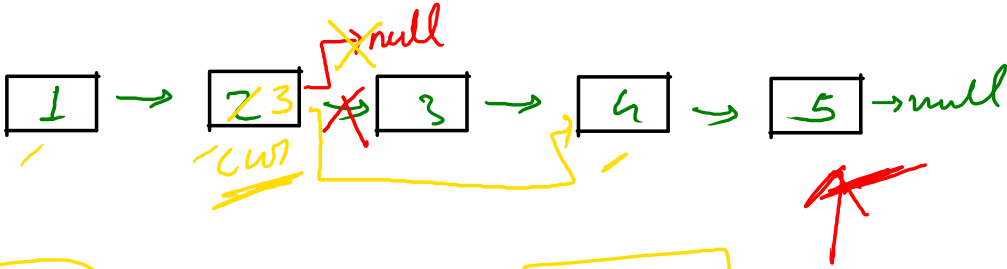
given → pointer cur to node '2'
& we need to delete it





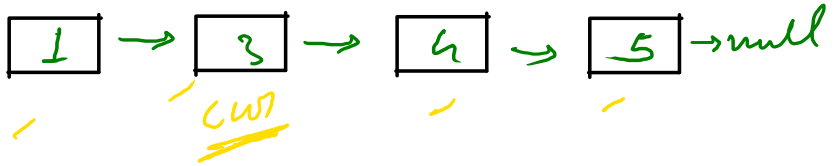
1. cur.data = cur.next.data
   cur.next = null



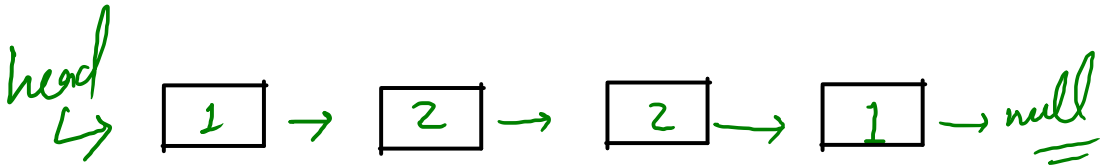


cur.data = cur.next.data
cur.next = ~~null~~ cur.next.next

o/p →



Palindrome List :-



Approach 1

→ make new LL (same)

→ & reverse it

→ compare the original & reversed one by one.

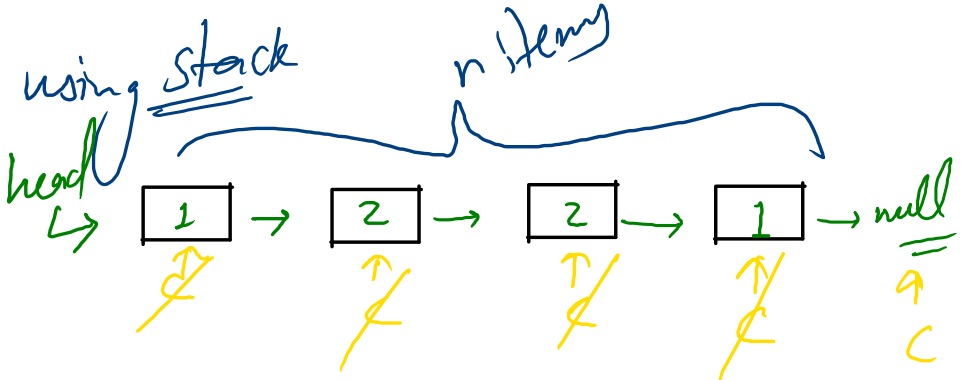
Approach 2

using stack.

Approach 3

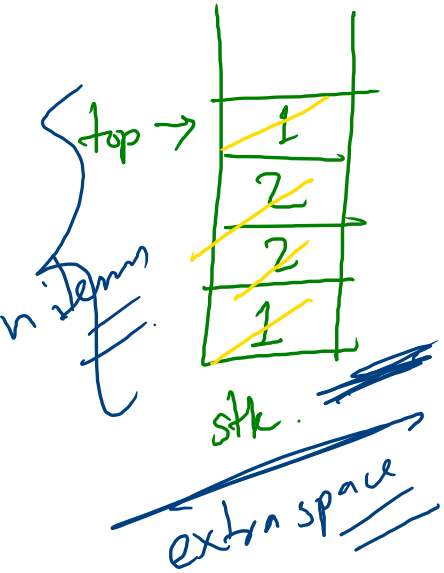
Two pointer approach

using stack



LIFO

```
c = head.
stk ← stack.
while( c != null)
{ stk.push(c.data)
  c = c.next
}
```



stk.peek()
stk.pop()

```
c = head
while( c != null)
{ if(c.data != stk.peak())
  { return false.
  }
  stk.pop()
  c = c.next
}
return true.
```

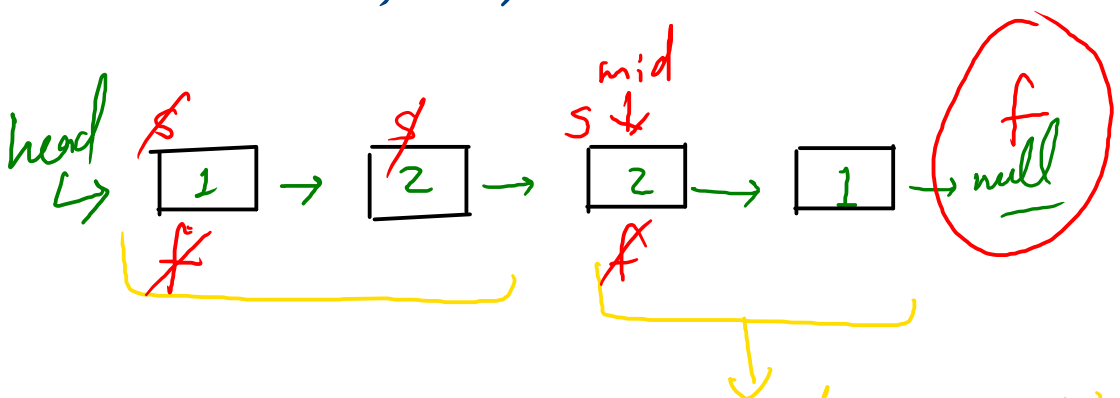
T.C. = O(n)

S.C. = O(n)

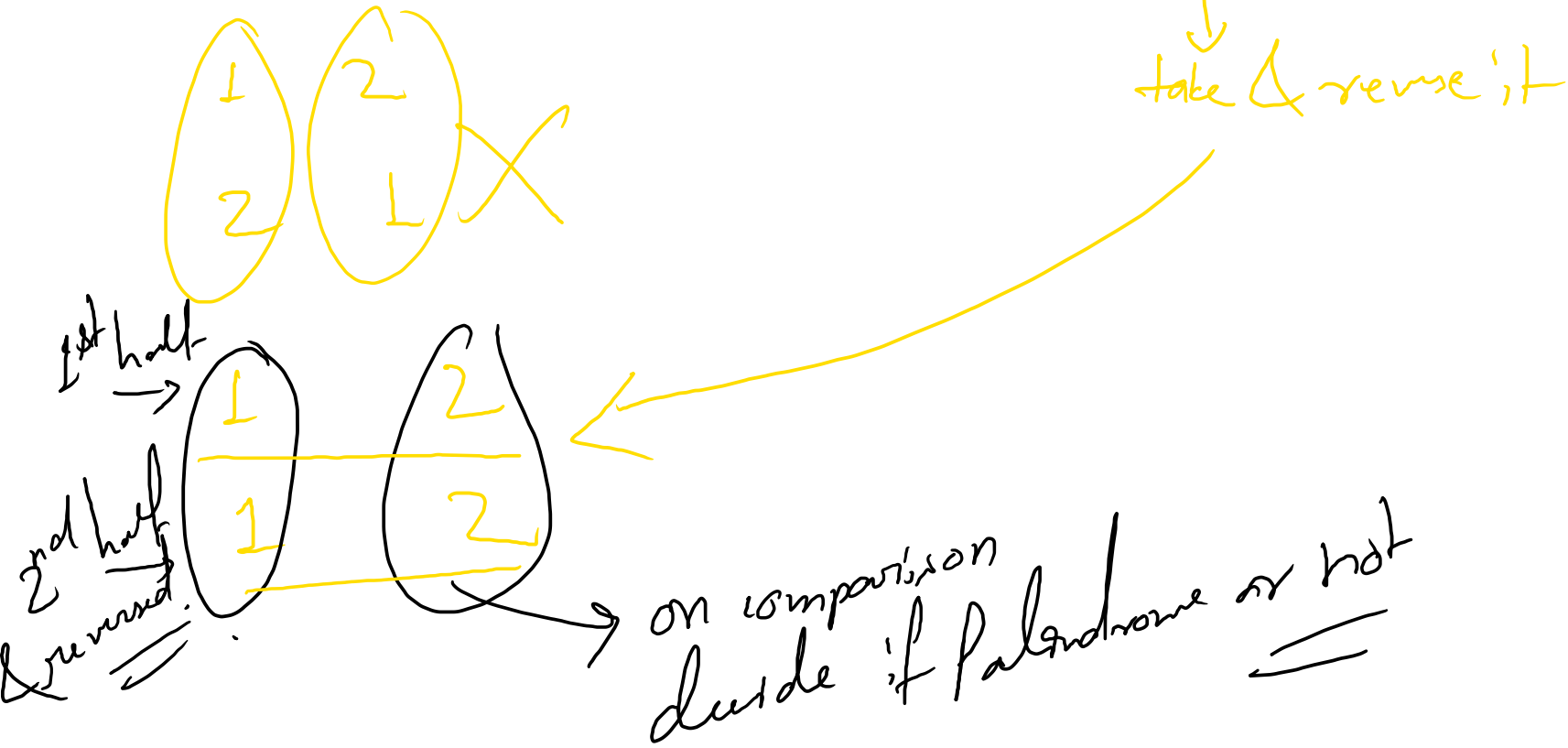
3rd Approach

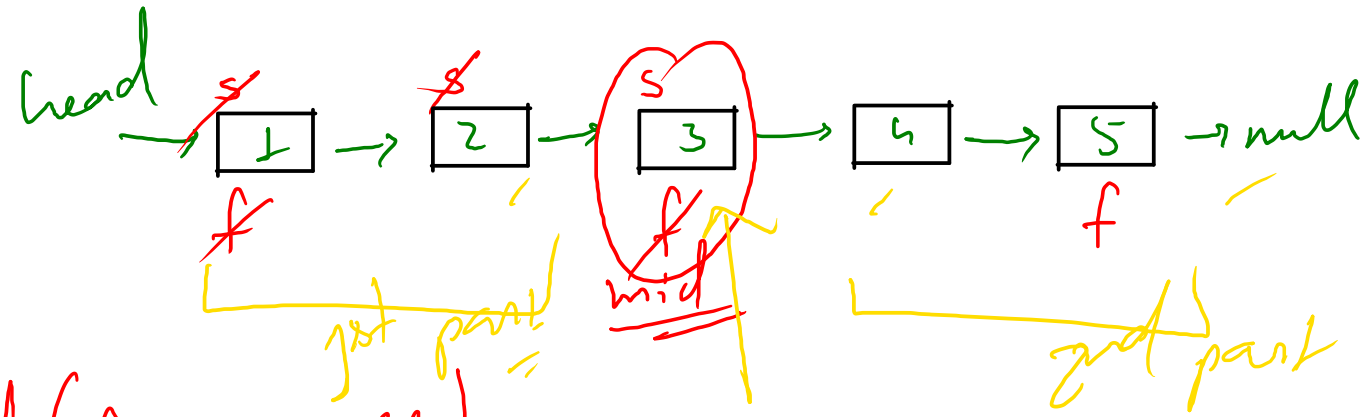
T.C. → O(n) , S.C. → O(1)

Steps



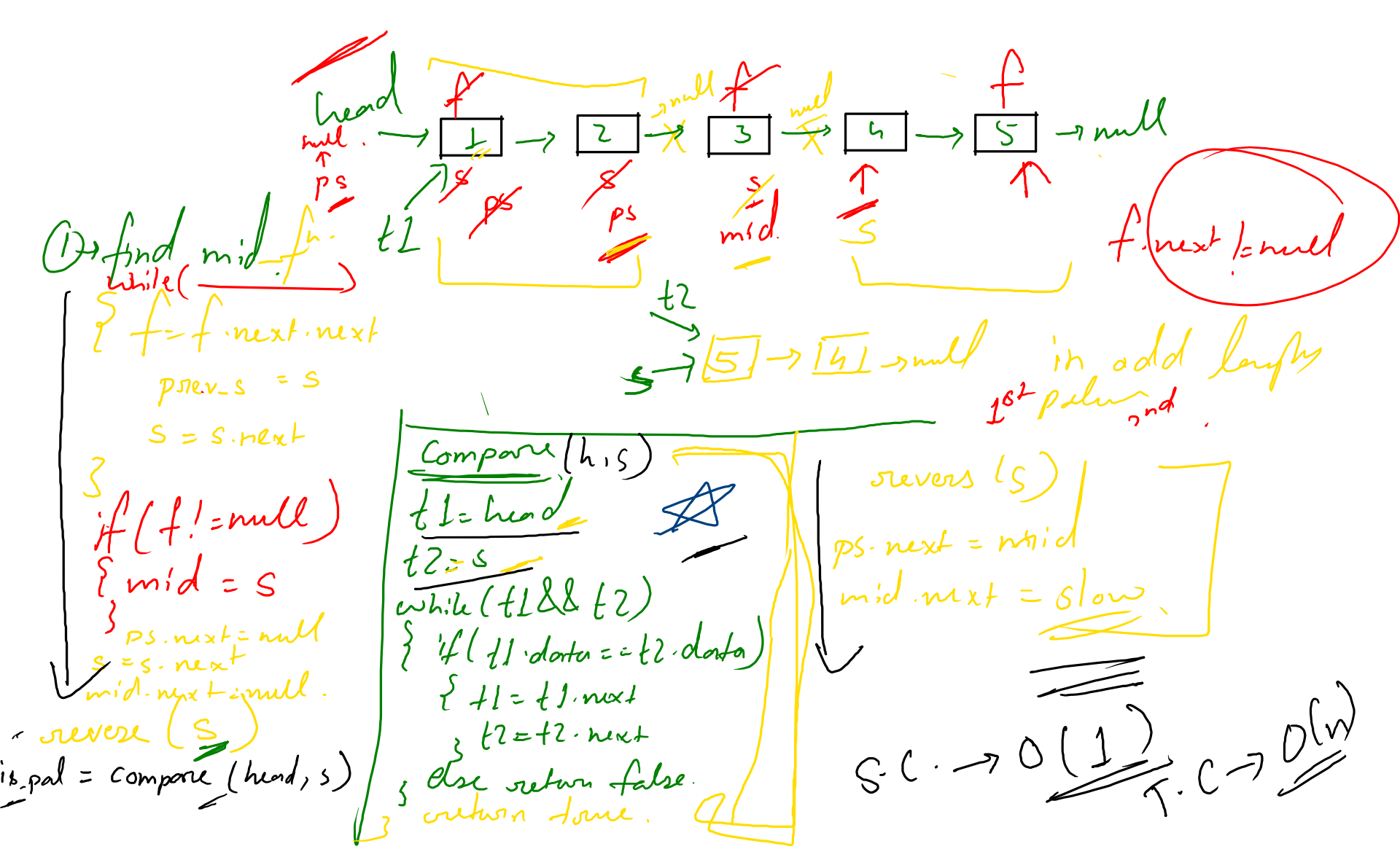
take & reverse it



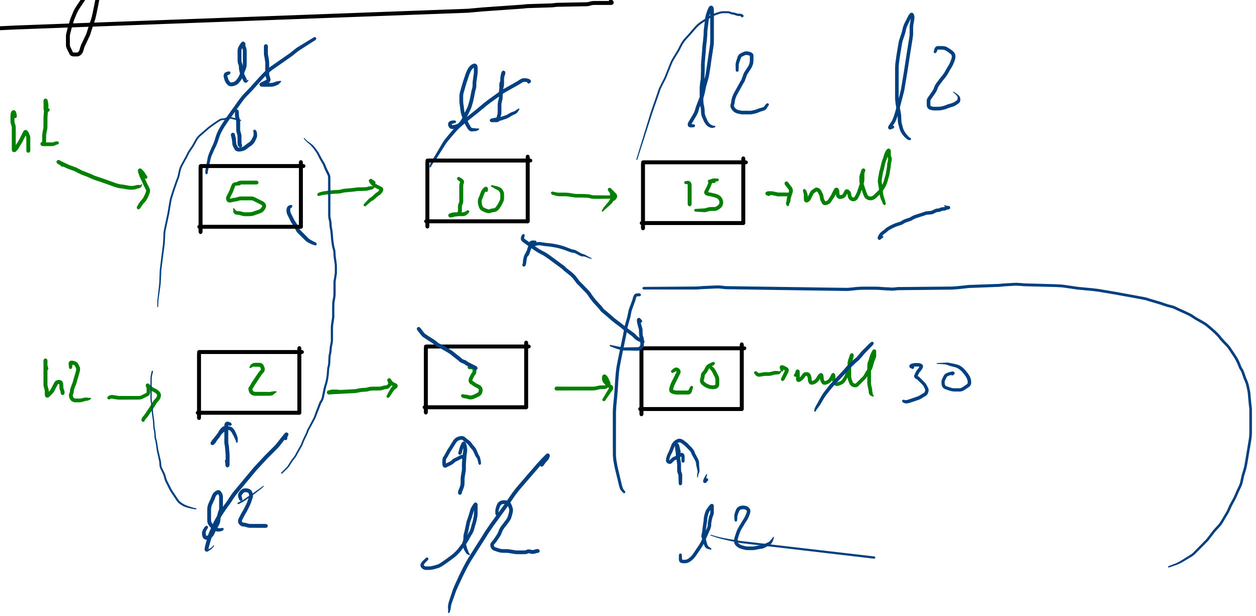


odd length

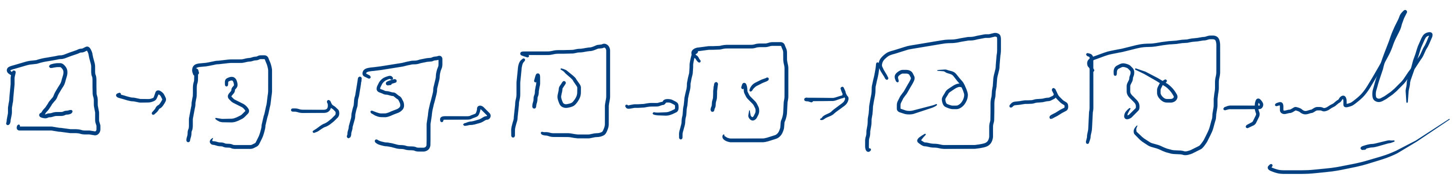
if (fast != null)
mean odd length.


head
1
2
3
4
5
null
ps
s
f
mid
t1
t2
f.next != null
1) find mid
while(
f = f.next.next
prev_s = s
s = s.next
if(f!=null)
mid = s
ps.next = null
s = s.next
mid.next = null
reverse (s)
is_pal = compare (head, s)
t2
5 → 4 → null
in odd length
1st palin
2nd
Compare (h, s)
t1 = head
t2 = s
while (t1 && t2)
if (t1.data == t2.data)
t1 = t1.next
t2 = t2.next
else return false.
return true.
revers (s)
ps.next = mid
mid.next = slow
S.C. → O(1)
T.C → O(n)

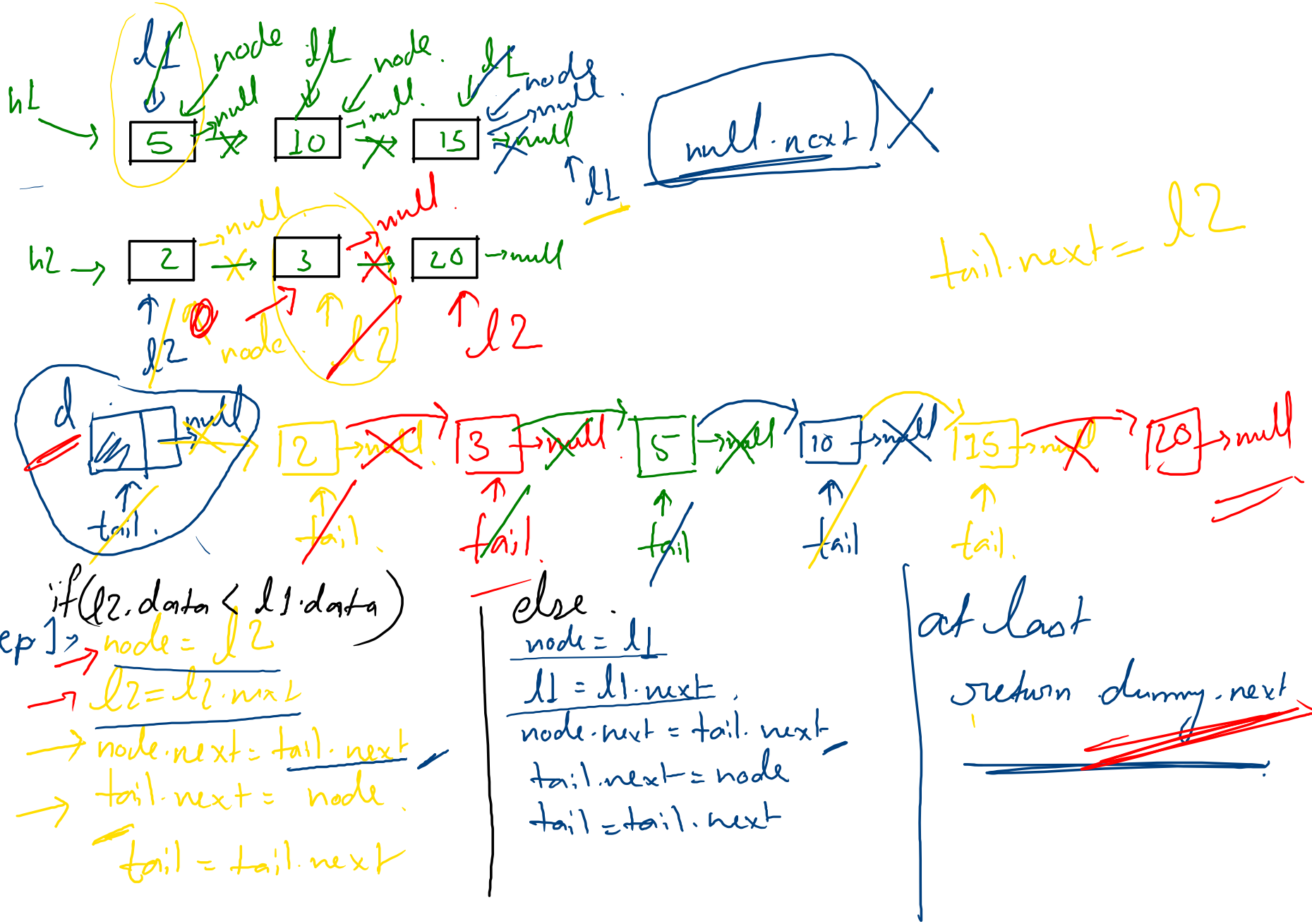
Merge Two LL :-



h1
l1
l1
l2
l2
5
10
15
→null
h2
2
3
20
→null
30
l2
l2
l2



2
3
5
10
15
20
30
→null


l1
node
l1
node
l1
node
h1
5
10
15
null
null
null
null
null.next
l1
h2
2
3
20
null
null
null
l2
l2
node
l2
tail.next = l2
d
null
tail
2
3
5
10
15
20
null
tail
tail
tail
tail
tail
if(l2.data < l1.data)
Step 1 >
node = l2
l2 = l2.next
node.next = tail.next
tail.next = node
tail = tail.next
else
node = l1
l1 = l1.next
node.next = tail.next
tail.next = node
tail = tail.next
at last
return dummy.next

```
merge(l1, l2)
{ dummy = new Node( )
  tail = dummy.
while( l1 || l2 )
{ if ( l1 == null )
  { tail.next = l2
    break.
  }
  if( l2 == null)
  { tail.next = l1
    break.
  }
if( l2.data < l1.data)
{ node = l2
    l2 = l2.next
    node.next = tail.next
   tail.next = node
}
else
{ node = l1
  l1 = l1.next
  node.next = tail.next
  tail.next = node.
}
tail = tail.next
}
return dummy.next
}
```


l2 = null.

T.C

l1 → n

l2 → m

↳ T.C → O(m+n)

↳ S.C → O(1)