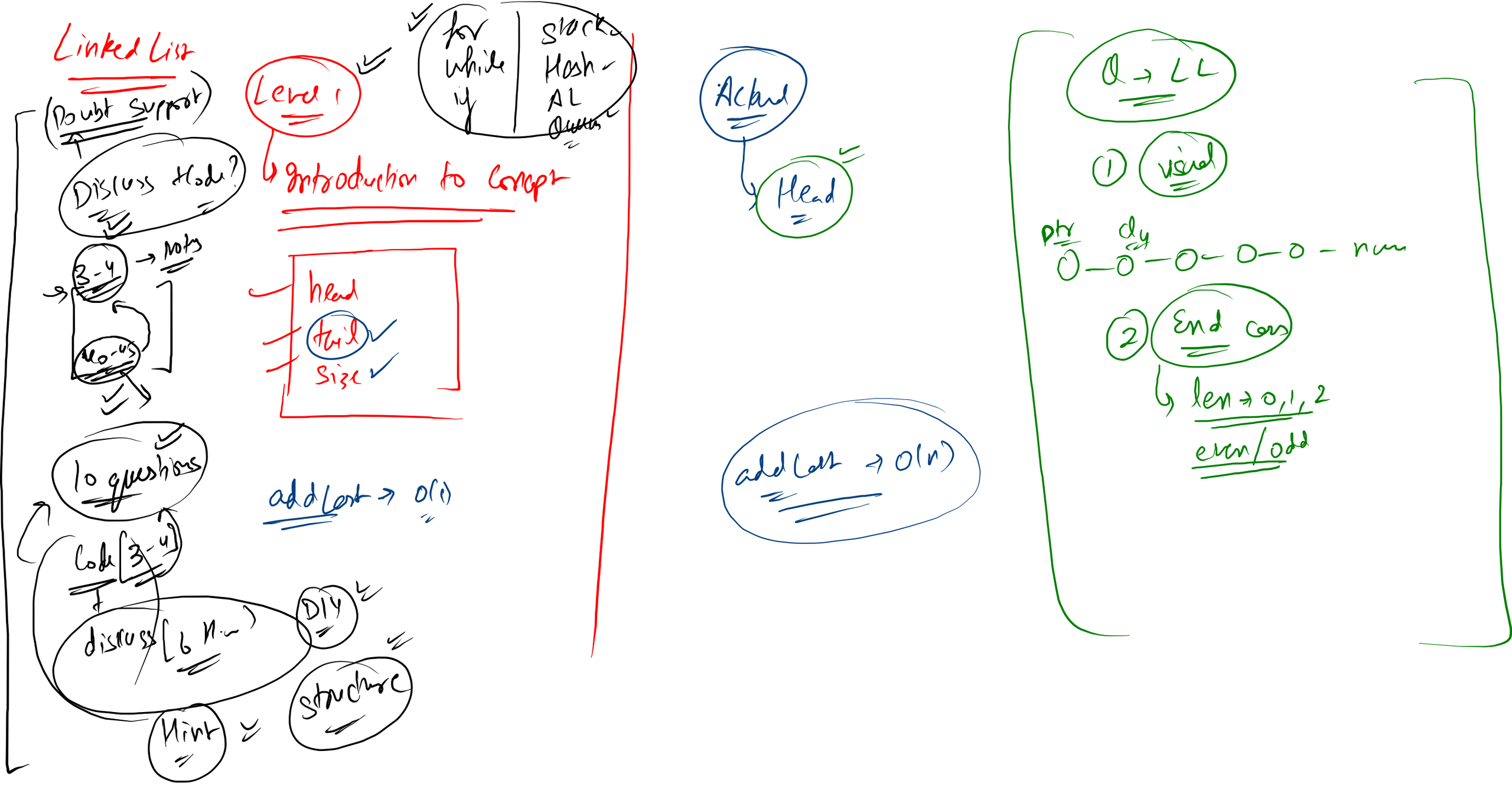
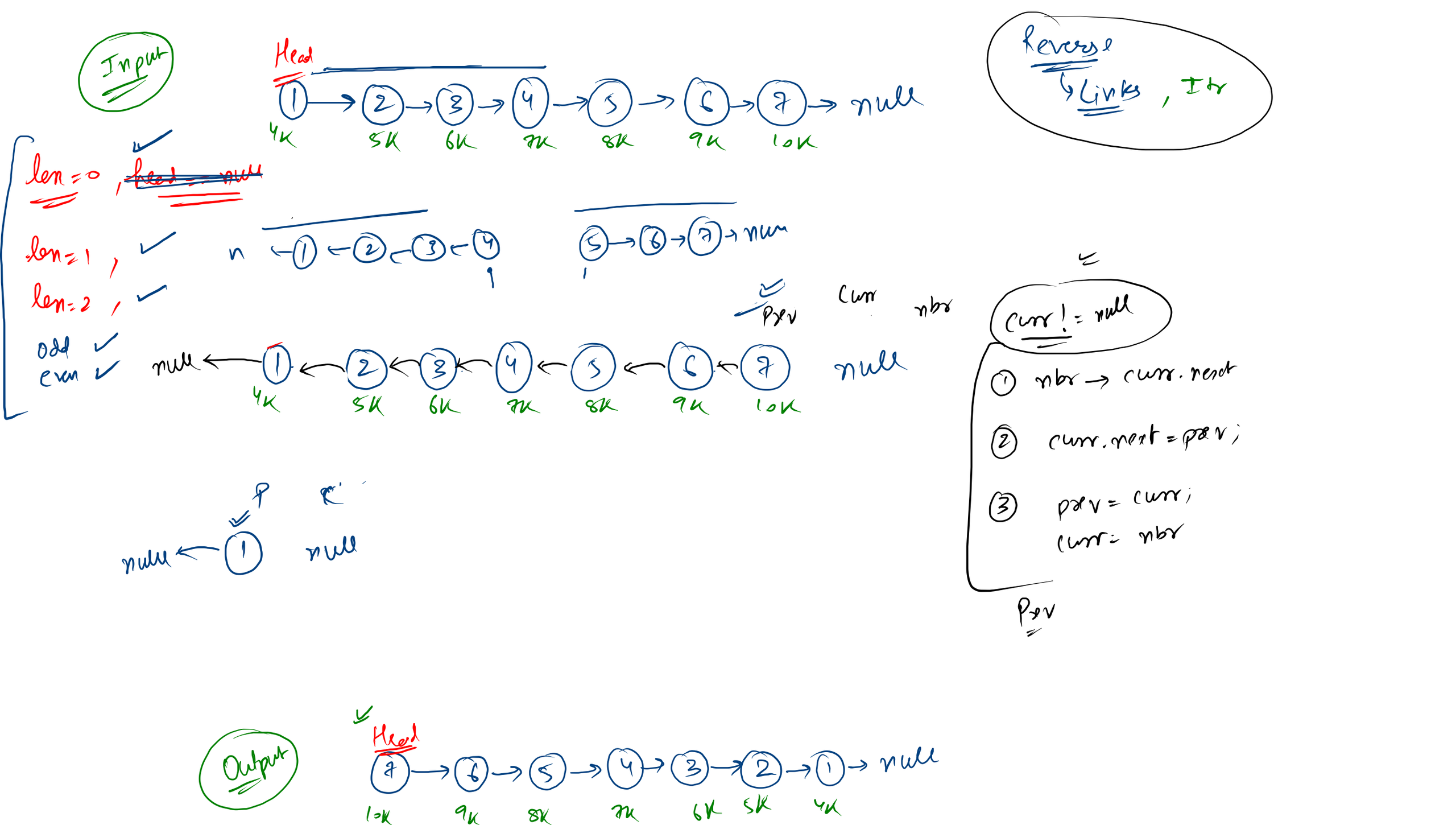
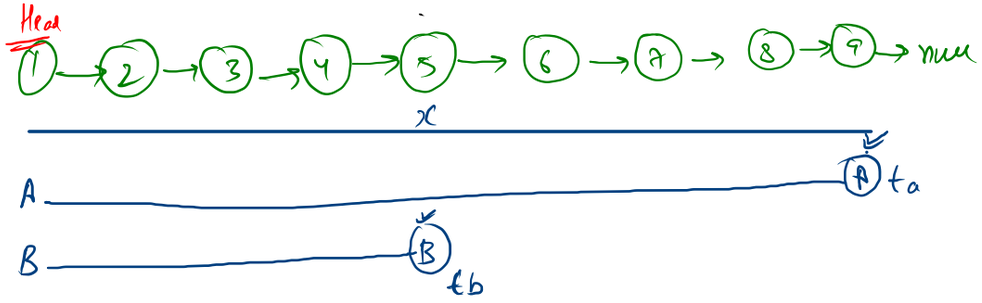

Linked List
Doubt Support
Discuss Hodi?
Level 1
for
while
if
stack
Hash -
AL
Introduction to Concept
3-4
Notes
head
tail
size
10 questions
addLast → O(1)
Code [3-4]
discuss [6 Min]
DIY
structure
Hint
Actual
Head
addLast → O(n)
Q → LL
1
visual
ptr
dy
2
End cons
len → 0, 1, 2
even/odd

</>	Reverse A Linkedlist	Easy	10	Auth	0	Public	Sol	1
</>	Middle Of A Linked List	Easy	10	Auth	0	Public	Sol	2
</>	Palindrome Linkedlist	Easy	10	Auth	0	Public	Sol	3
</>	Fold Of Linkedlist	Easy	10	Auth	0	Public	Sol	4
</>	Unfold Of Linkedlist	Easy	10	Auth	0	Public	Sol	5
</>	Merge Two Sorted Linkedlist	Easy	10	Auth	0	Public	Sol	6
</>	Mergesort Linkedlist	Easy	10	Auth	0	Public	Sol	7
</>	Remove Nth Node From End Of Linkedlist	Easy	10	Auth	0	Public	Sol	8
</>	Add Two Linkedlist	Easy	10	Auth	0	Public	Sol	9
</>	Subtract Two Linkedlist	Easy	10	Auth	0	Public	Sol	10
</>	Multiply Two Linkedlist	Easy	10	Auth	0	Public	Sol	11


Input
Head
1 → 2 → 3 → 4 → 5 → 6 → 7 → null
4K 5K 6K 7K 8K 9K 10K
Reverse
Links , Itr
len = 0 , head = null
len = 1 ,
len = 2 ,
odd
even
n
1 ← 2 ← 3 ← 4
5 → 6 → 7 → null
prev
curr
nbr
null ← 1 ← 2 ← 3 ← 4 ← 5 ← 6 ← 7
null
curr != null
① nbr → curr.next
② curr.next = prev;
③ prev = curr;
curr = nbr
prev
p
k
null ← 1
null
Output
Head
7 → 6 → 5 → 4 → 3 → 2 → 1 → null
10K 9K 8K 7K 6K 5K 4K

odd
even



Fast.next == null

Slow → mid

Speed = distance / time

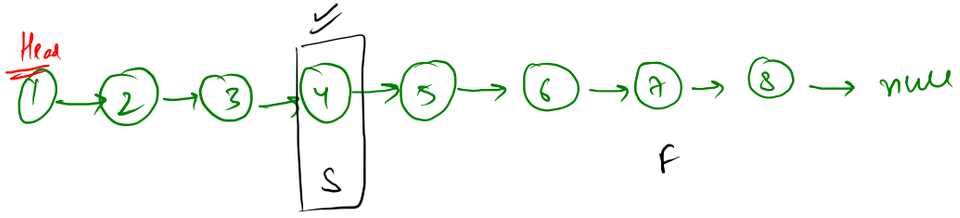
$S_a = \frac{x}{t_a}$, $S_b = \frac{x}{2t_b}$

$t_a = \frac{x}{S_a}$, $t_b = \frac{x}{2S_b}$

$t_a = t_b$

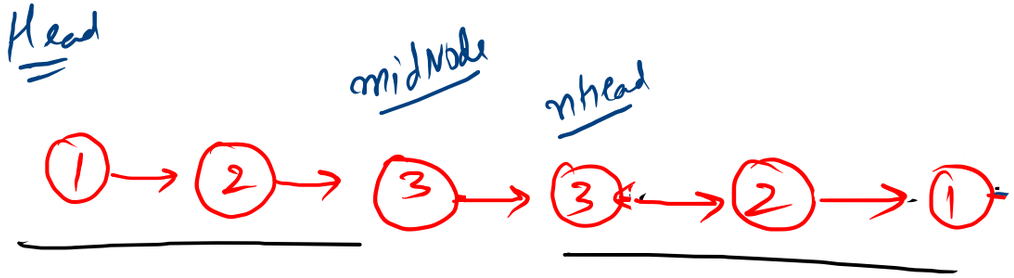
$\frac{x}{S_a} = \frac{x}{2S_b} \Rightarrow S_a = 2S_b$

even



fast.next.next == null

slow → mid



1. midNode [n]
2. nHead [1]
3. delink [1]
4. reverse 2nd part [n/2]
5. check palindrome → as T [n/2]
6. rebuild your linkList
   - return 2nd list [n/2]
   - relink [1]
7. ans

$$T = n + 1 + 1 + \frac{n}{2} + \frac{n}{2} + \frac{n}{2} + 1$$

$$T \Rightarrow \frac{5n}{2} + 3$$

$$T \Rightarrow 2.5\,n + 3 \Rightarrow O(n)$$

$$S \Rightarrow O(1)$$

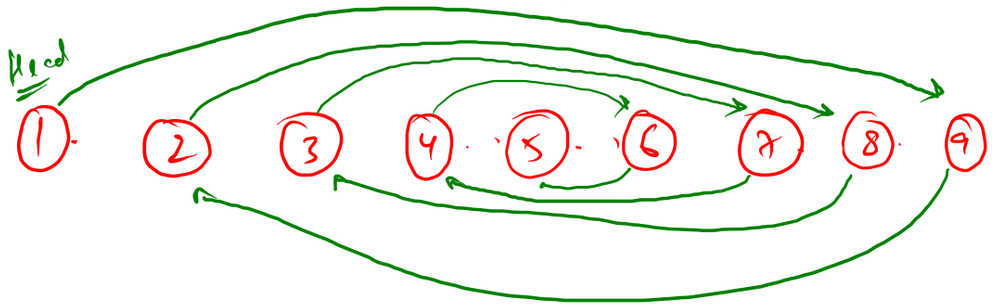
```java
public static boolean isPalindrome(ListNode head) {
    ListNode midNode = middleOFLL(head);
    ListNode nhead = midNode.next;
    midNode.next = null;

    nhead = reverse(nhead);

    ListNode p1 = head , p2 = nhead;
    boolean res = true;
    while(p2 != null){
        if(p1.val != p2.val){
            // non palindromic
            res = false;
            break;
        }

        p1 = p1.next;
        p2 = p2.next;
    }

    nhead = reverse(nhead);
    midNode.next = nhead;

    return res;
}
```

ans = T

Head

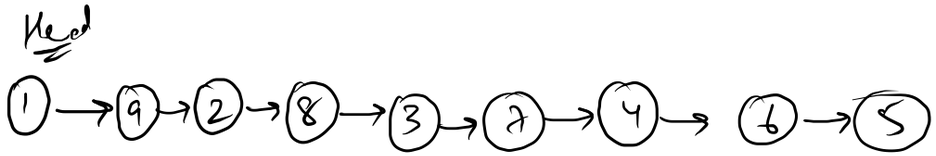
Input :

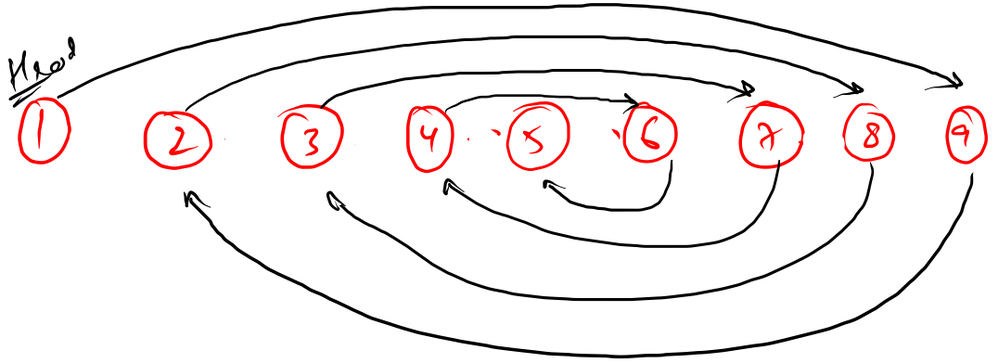
reverse

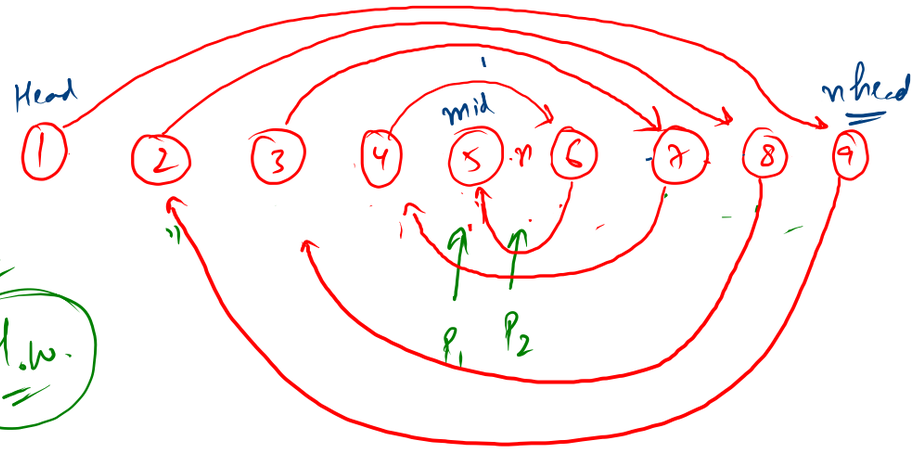


p_2 p_1

Output :-





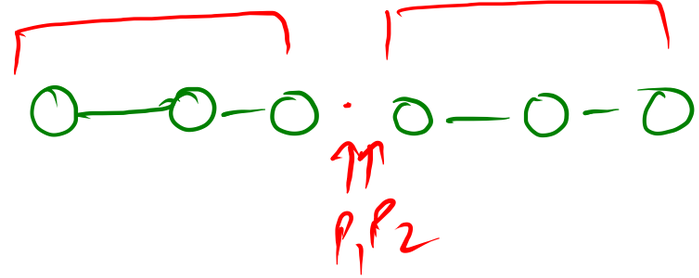


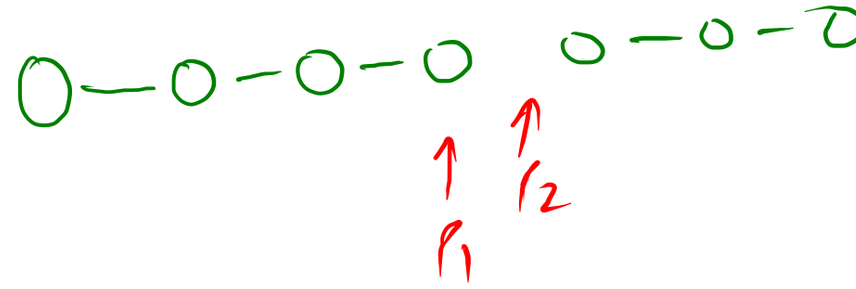
H.W.

∴ p_1 != null && p_2 != null

∴ p_2 != null

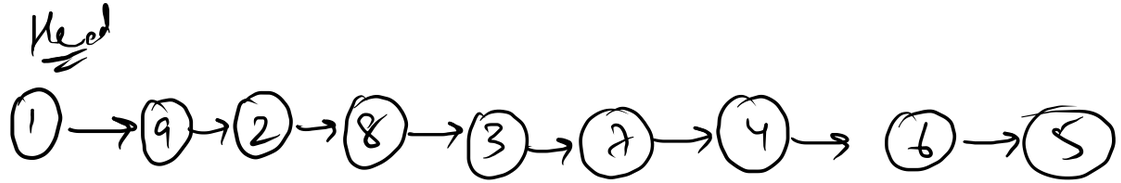
```
while (p2 != null)
    tp1 -> p1.next
    tp2 -> p2.next
    p1.next = p2
    p2.next = tp1
    p1 = tp1
    p2 = tp2
return head;
```



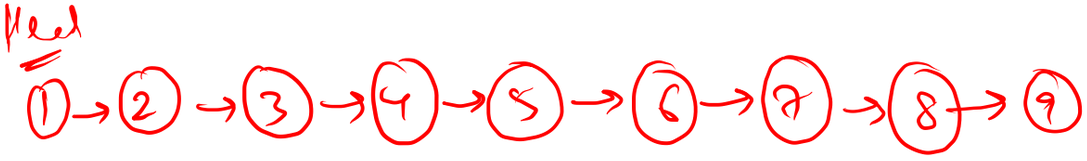


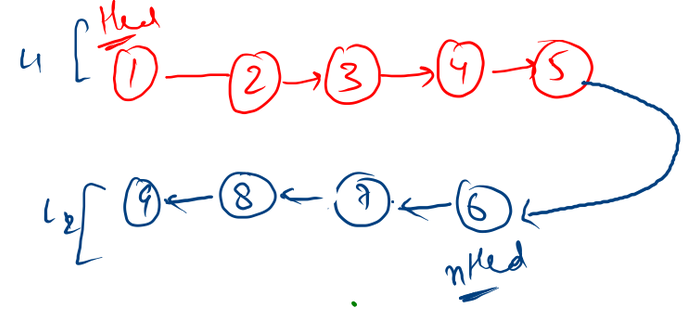
Unfold

Input



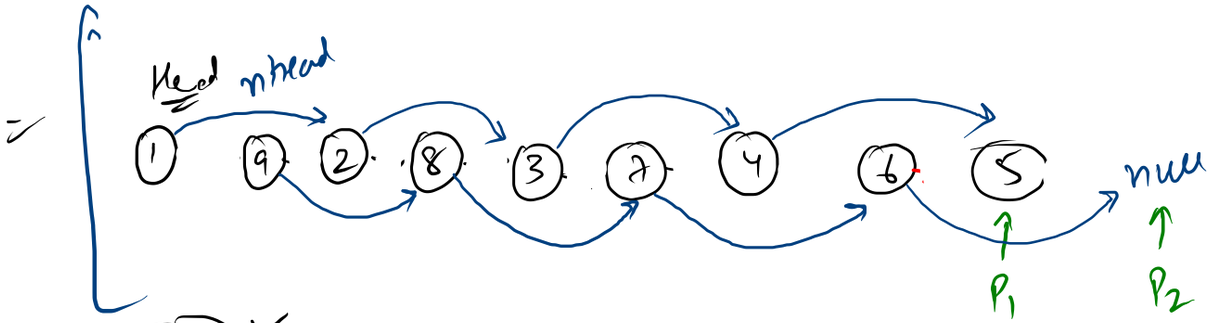
Output

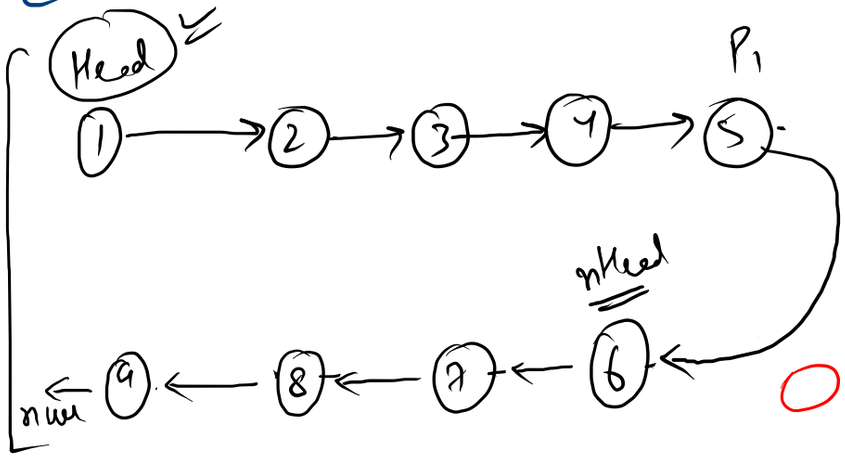


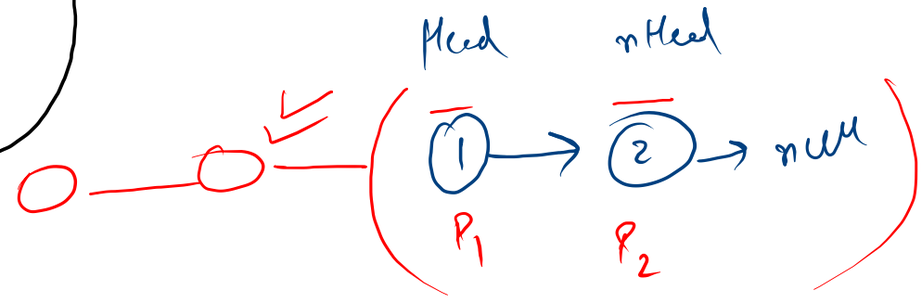


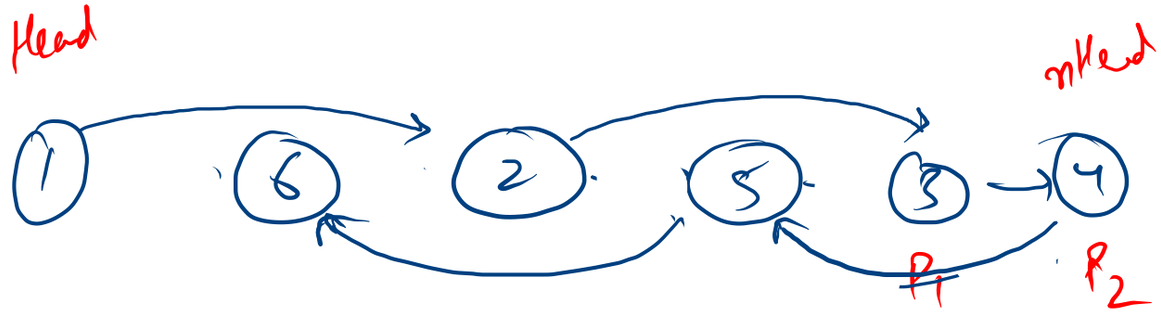

```
P1.next = P1.next.next
P2.next = P2.next.next
P1 = P1.next
P2 = P2.next
```

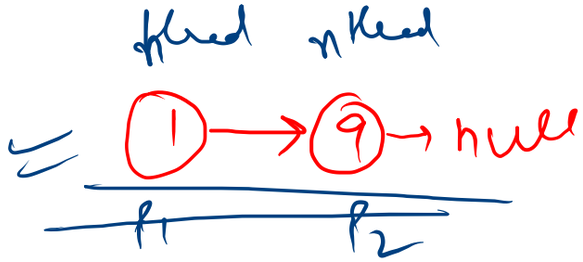
$P_2 != null \;\&\&\; P_2.next != null$



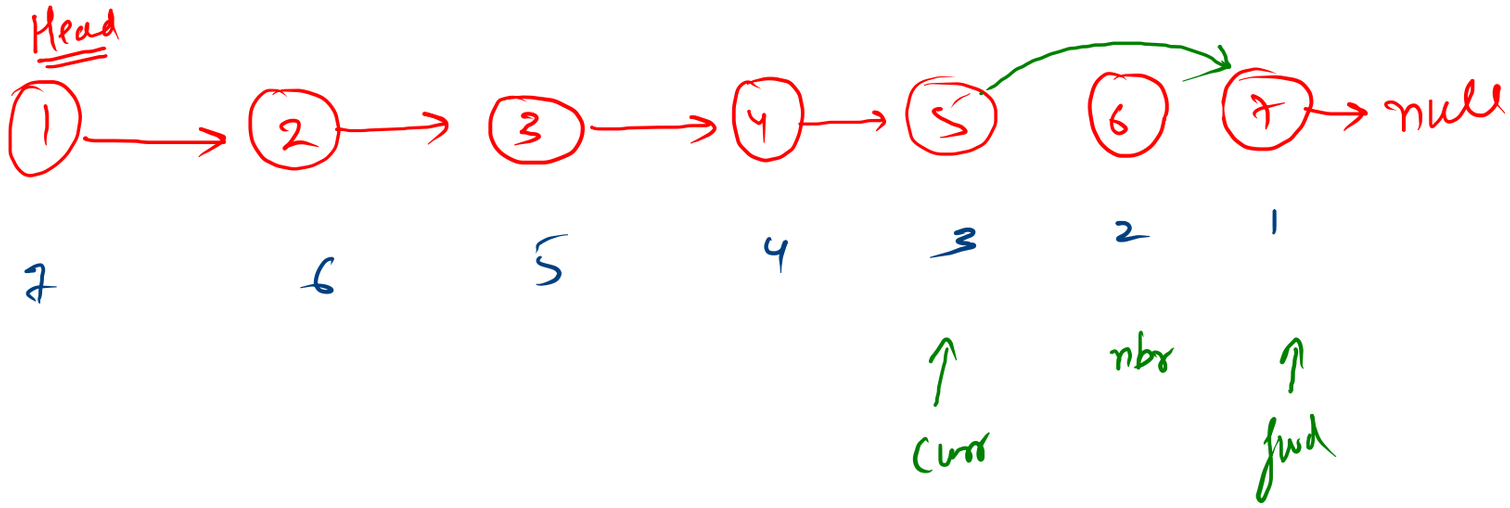





Head
nHead
1
6
2
5
3
4
R_1
R_2


head
nHead
1
9
null
p1
p2

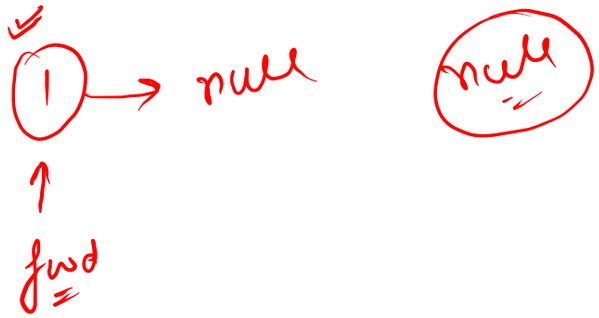
I



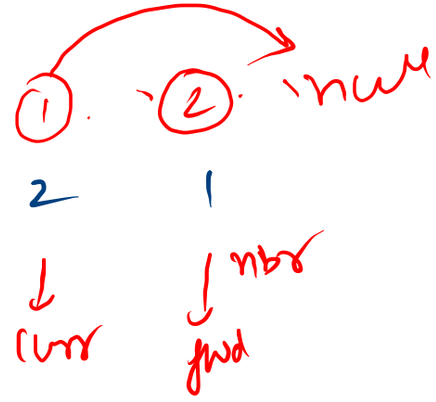
K = 2

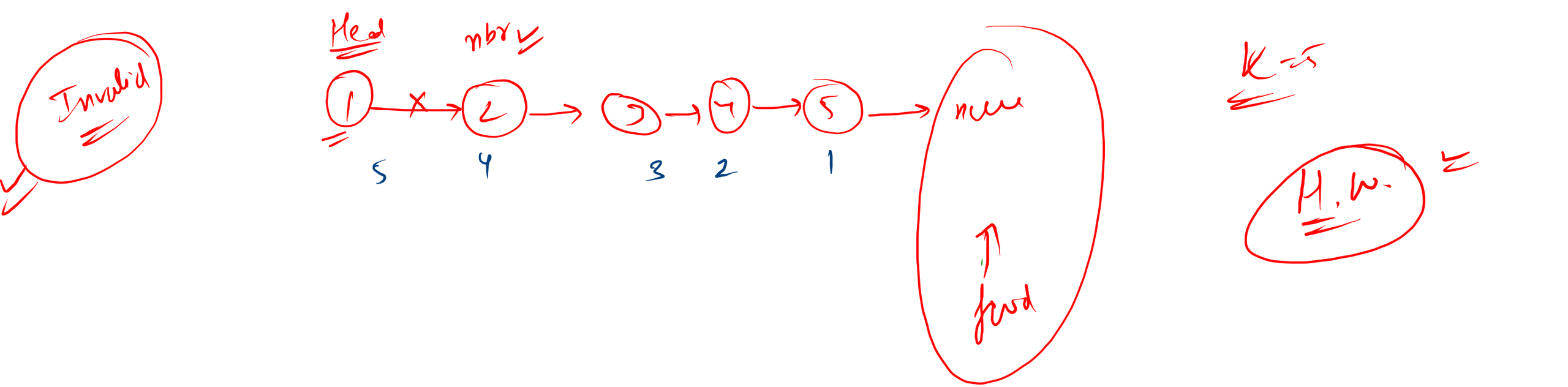
K^th node from the end

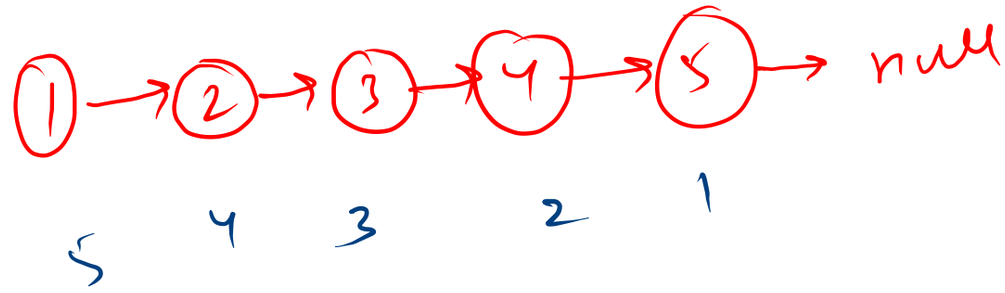
fwd.next != null

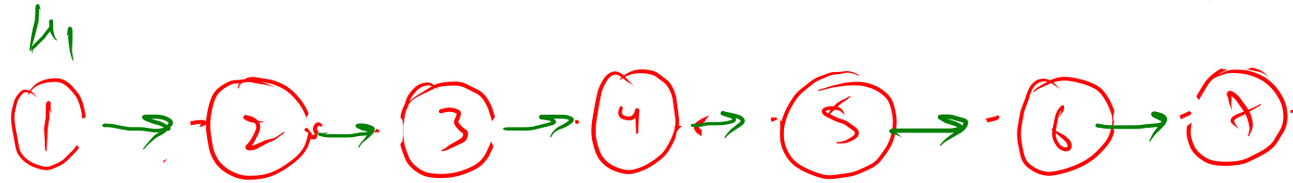


null



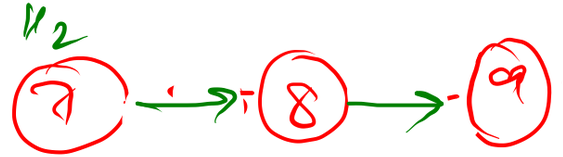

Invalid
Head
nbr
1
2
3
4
5
null
fwd
5
4
3
2
1
K = 5
H.W.


1
2
3
4
5
null
5
4
3
2
1

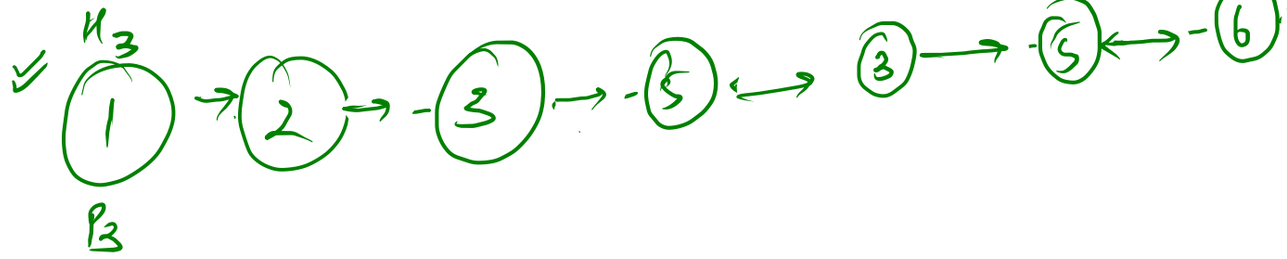


P_1

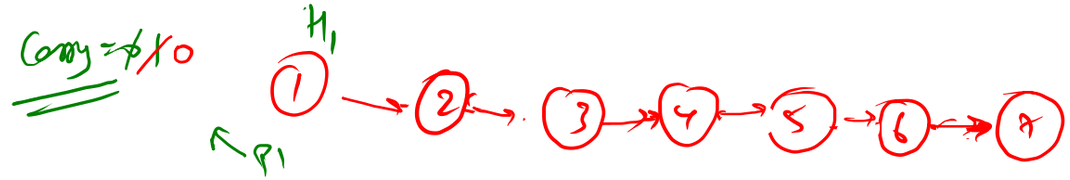
Carry = 0

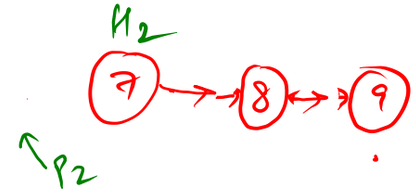


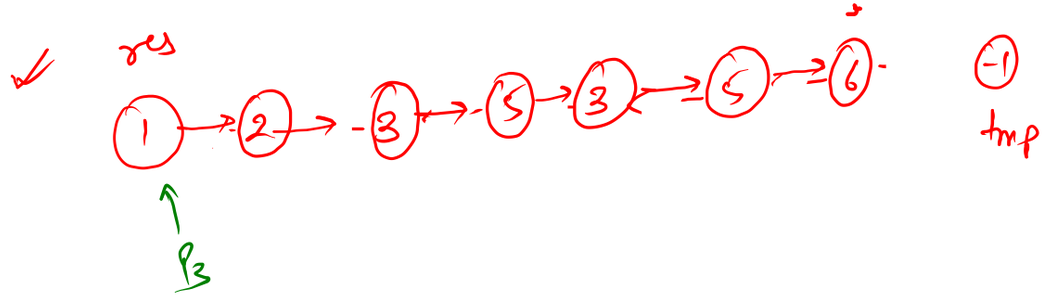
P_2



P_3







H1 H2

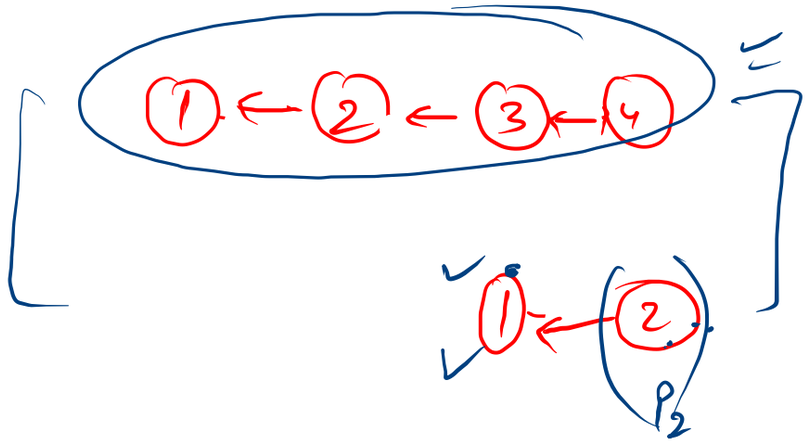
null null null

null — H2

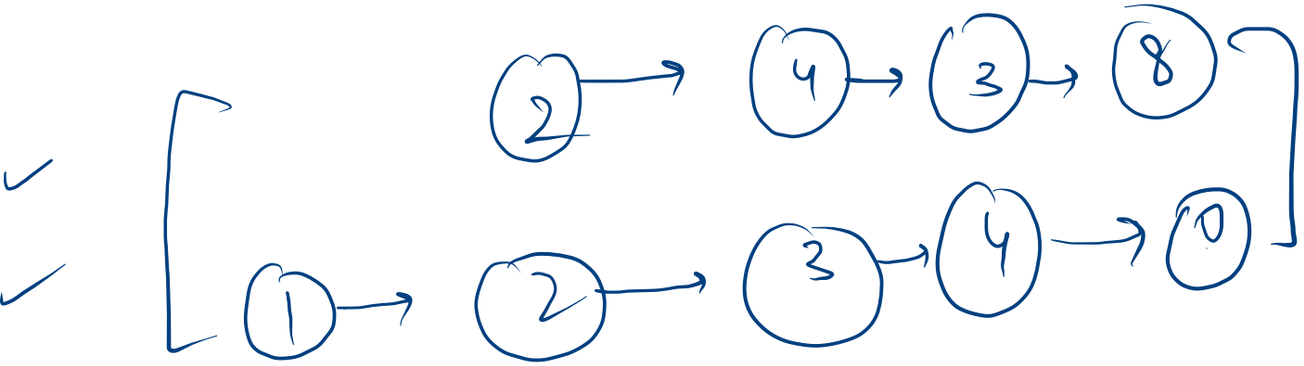
— null H1

— —

```java
public static ListNode addTwoNumbers(ListNode h1, ListNode h2) {
    if(h1 == null || h2 == null){
        return h1 == null ? h2 : h1;
    }

    h1 = reverse(h1);
    h2 = reverse(h2);

    ListNode res = new ListNode(-1);
    ListNode p1 = h1 , p2 = h2 , p3 = res;
    int carry = 0;

    while(carry != 0 || p1 != null || p2 != null){
        int v1 = (p1 == null) ? 0 : p1.val;
        int v2 = (p2 == null) ? 0 : p2.val;

        int sum = v1 + v2 + carry;

        carry = sum/10;
        sum = sum % 10;

        ListNode node = new ListNode(sum);
        p3.next = node;

        p3 = p3.next;
        p1 = (p1 == null) ? null : p1.next;
        p2 = (p2 == null) ? null : p2.next;
    }
    ListNode tmp = res;
    res = res.next;
    tmp.next = null;

    h1 = reverse(h1);
    h2 = reverse(h2);
    res = reverse(res);

    return res;
}
```

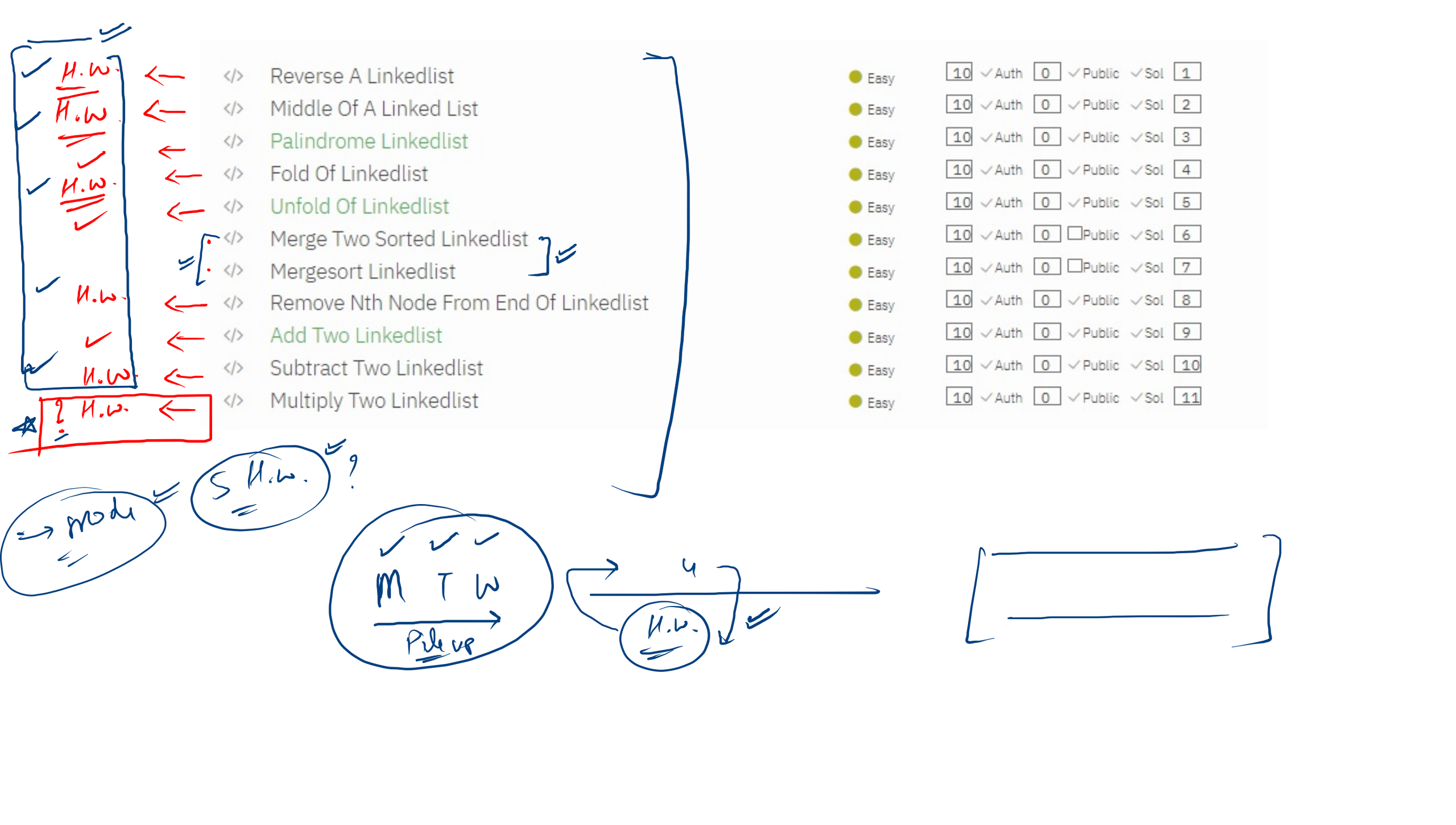


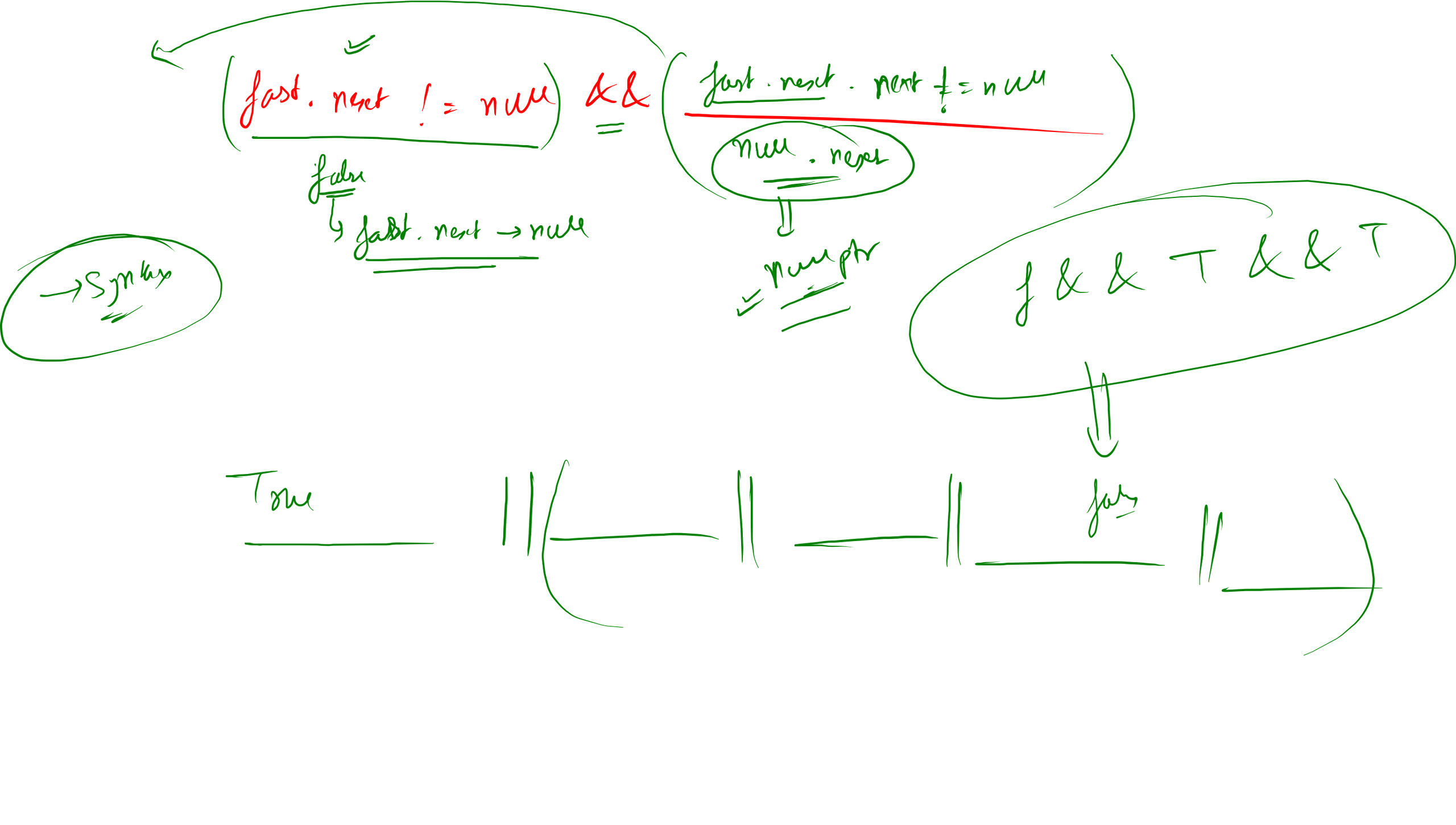
- ✓ addition
- ✓ Subtraction (H.W.)
- ? Multiplication



zero

vs


H.W.
H.W.
H.W.
H.W.
H.W.
? H.W.
Reverse A Linkedlist
Middle Of A Linked List
Palindrome Linkedlist
Fold Of Linkedlist
Unfold Of Linkedlist
Merge Two Sorted Linkedlist
Mergesort Linkedlist
Remove Nth Node From End Of Linkedlist
Add Two Linkedlist
Subtract Two Linkedlist
Multiply Two Linkedlist
Easy 10 Auth 0 Public Sol 1
Easy 10 Auth 0 Public Sol 2
Easy 10 Auth 0 Public Sol 3
Easy 10 Auth 0 Public Sol 4
Easy 10 Auth 0 Public Sol 5
Easy 10 Auth 0 Public Sol 6
Easy 10 Auth 0 Public Sol 7
Easy 10 Auth 0 Public Sol 8
Easy 10 Auth 0 Public Sol 9
Easy 10 Auth 0 Public Sol 10
Easy 10 Auth 0 Public Sol 11
mod
S H.W. ?
M T W
Pick up
4
H.W.


(fast.next != null) &&
fast.next.next != null
fals
fast.next → null
null.next
nullptr
→ Syntax
f && T && T
True
||
||
||
fals
||