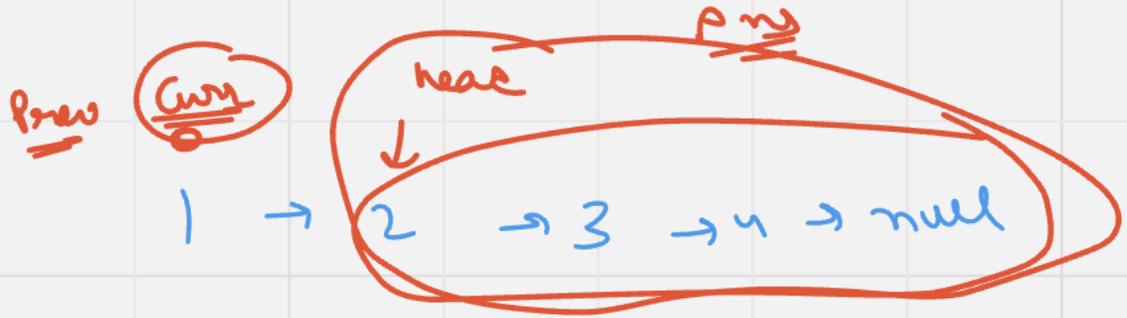
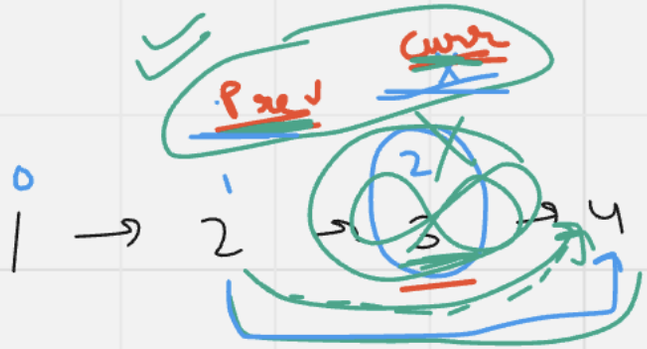


0

edge case

[head = curr.next

Problems on LL



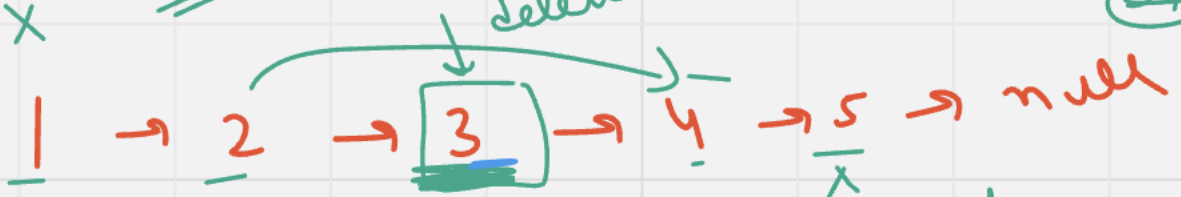
idx = 2

Correct
1 → 2 → 4 → null

[Prev.next = curr.next

head is not given

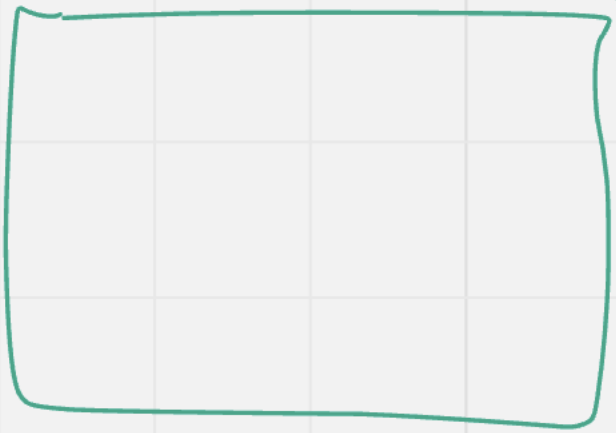
delete



not last

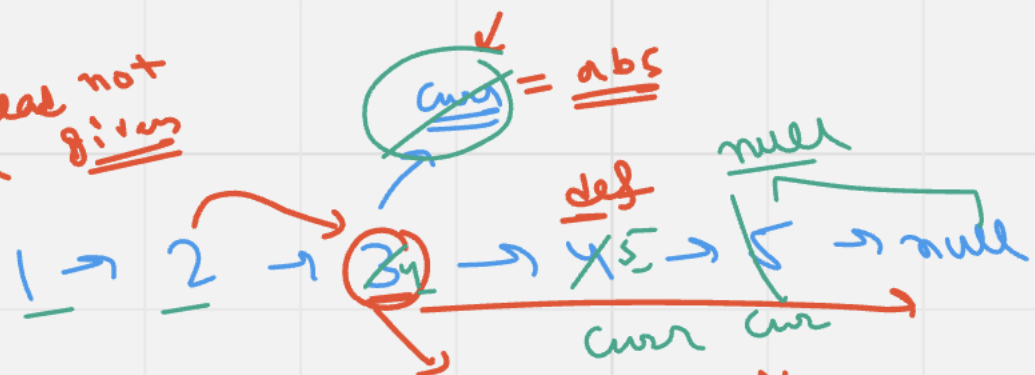
fun(Node curr)

unique



head not given

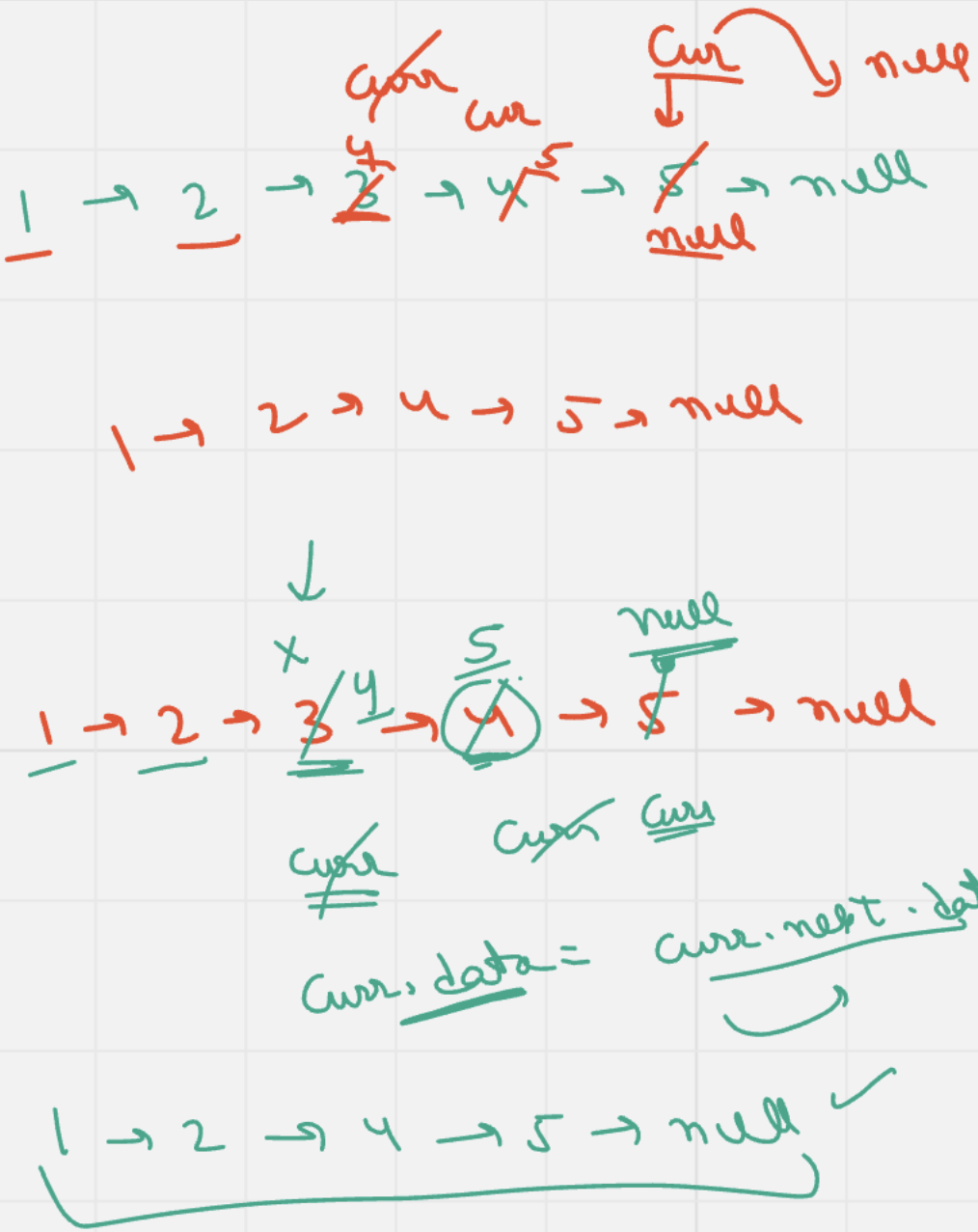
curr = abs

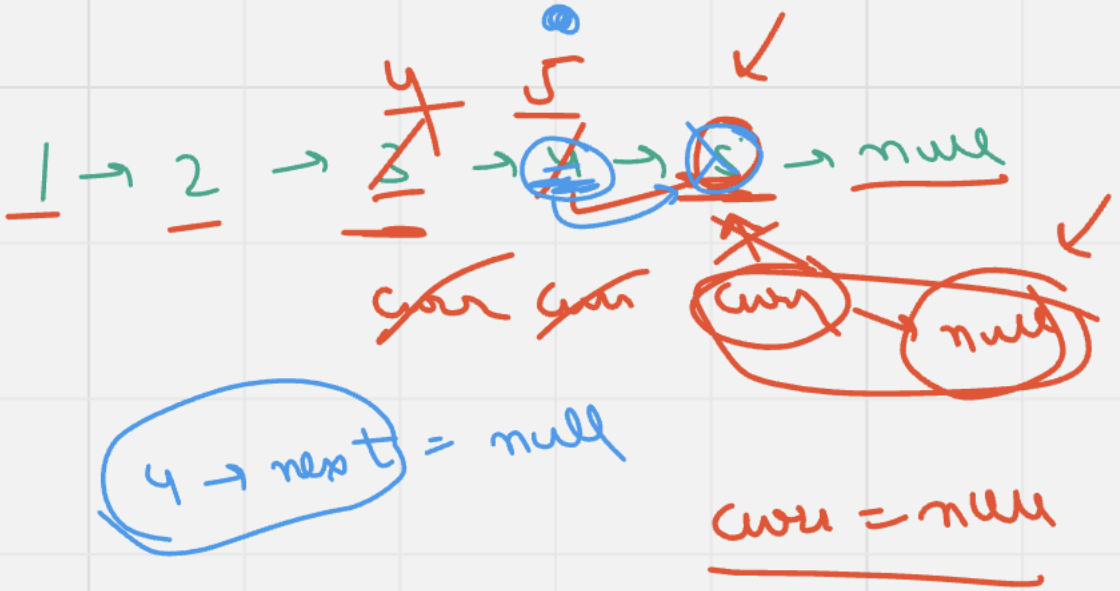


while tmp != null

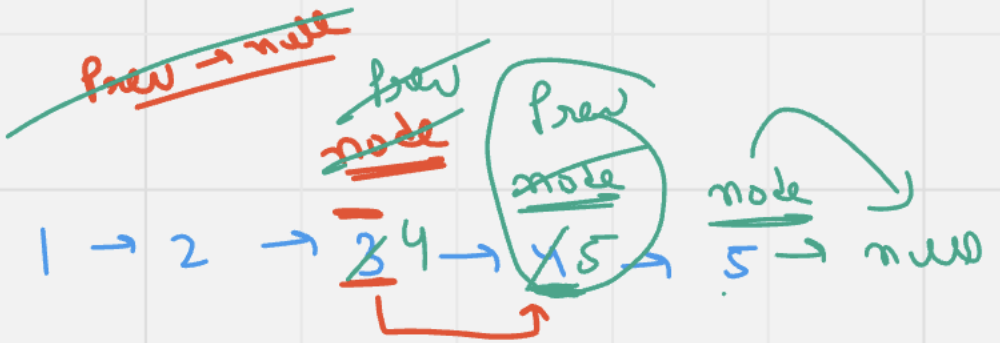
unique

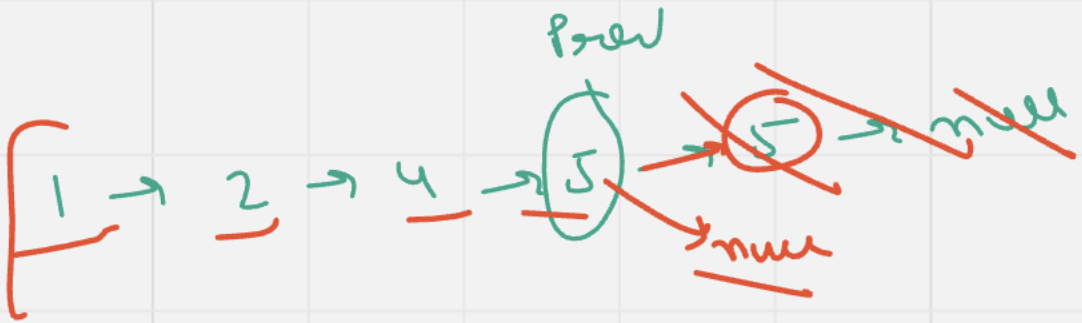
curr.val = curr.next.val

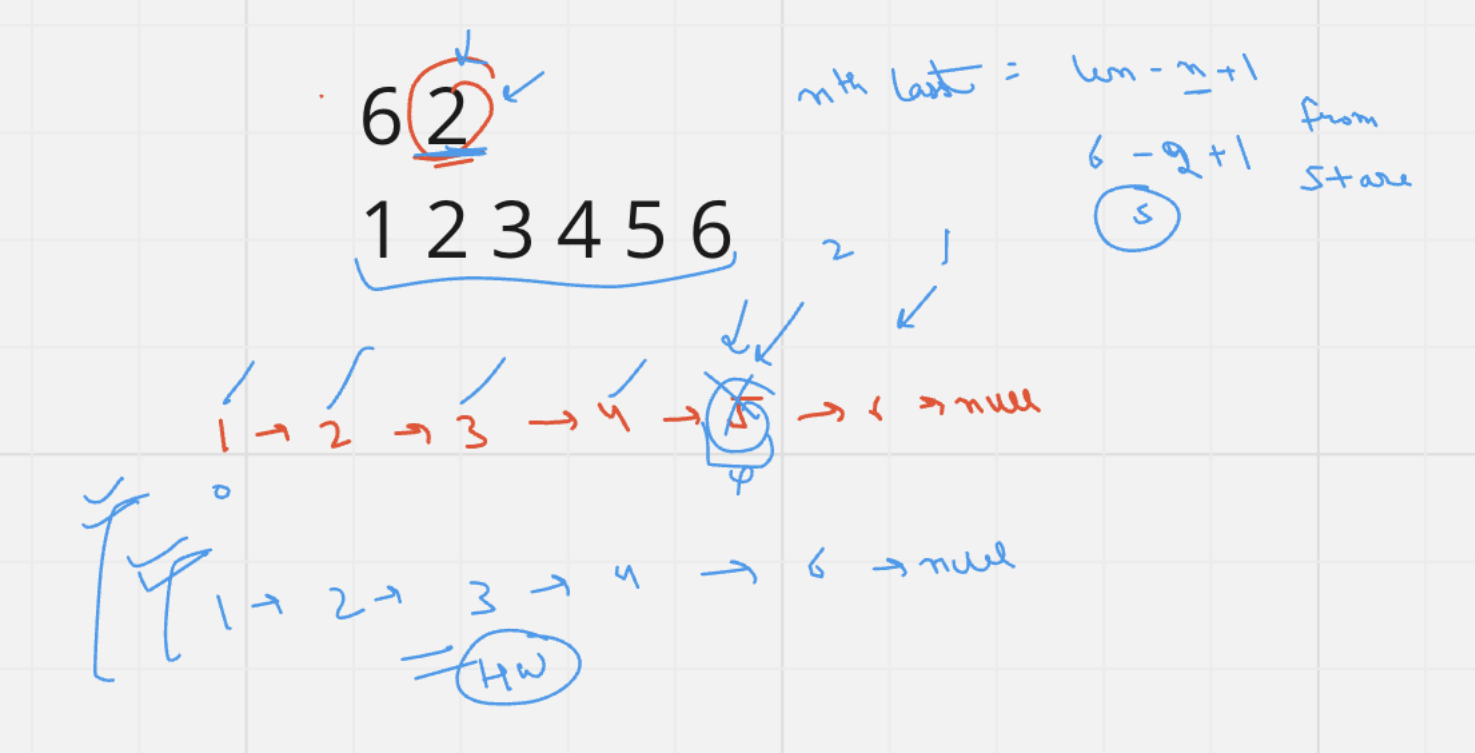

curr
curr
Curr
null
1 → 2 → 3 → 4 → 5 → null
null
1 → 2 → 4 → 5 → null
X
5
null
1 → 2 → 3 → 4 → 5 → null
curr
curr
curr
curr.data = curr.next.data
1 → 2 → 4 → 5 → null

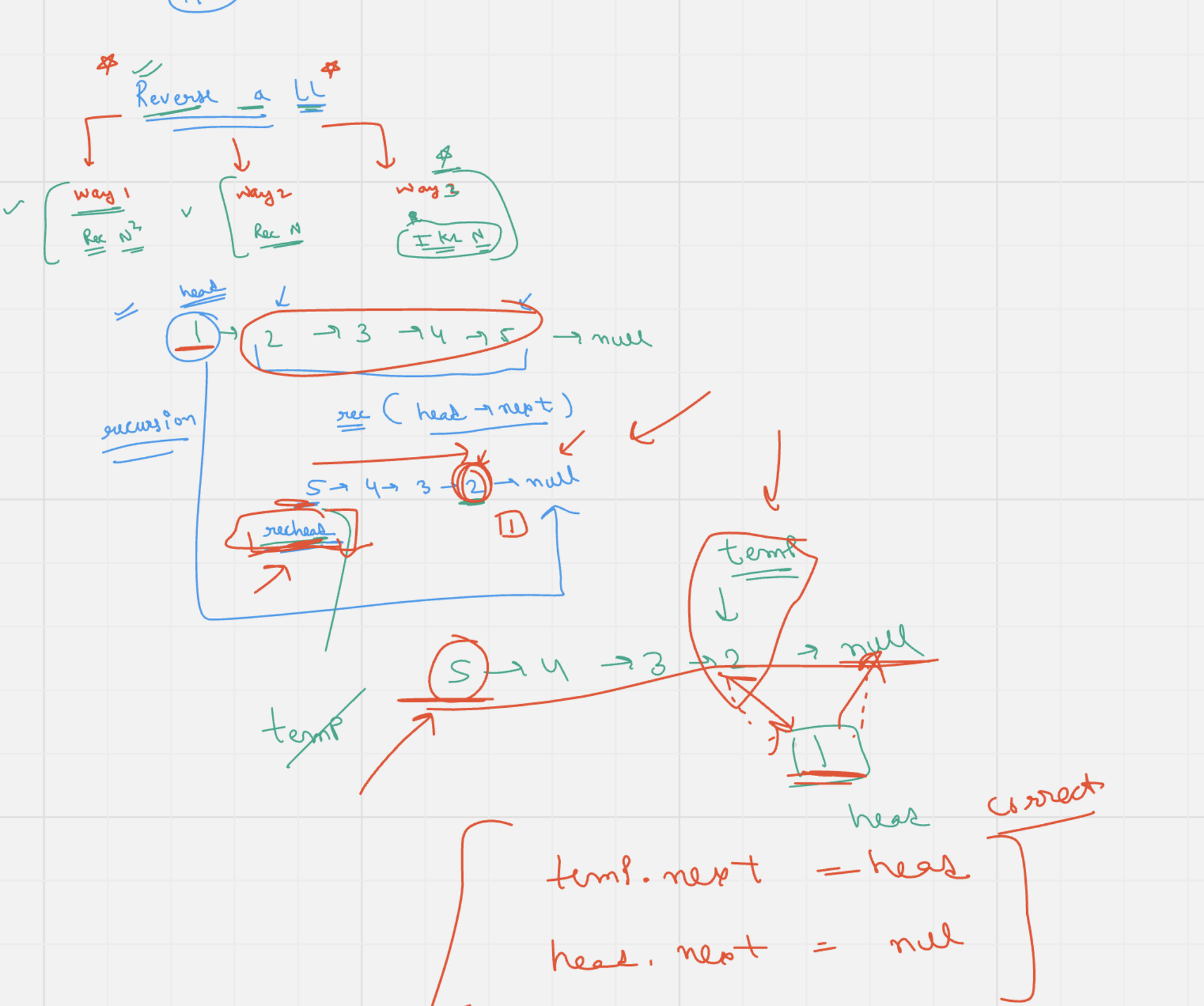


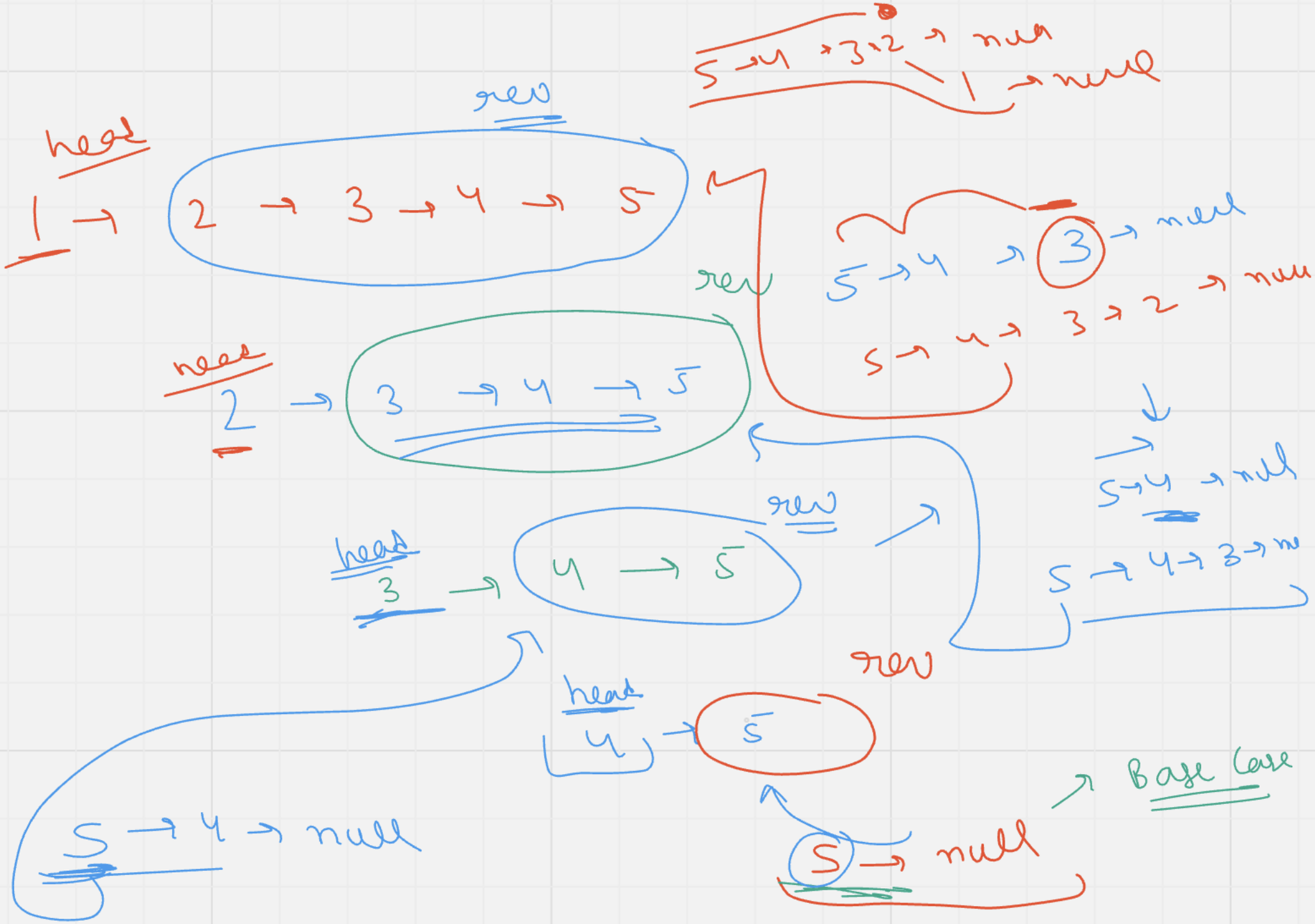

```
void deleteNode(Node node)
  {
     Node prev = node;
     while(node.next != null)
     {
        node.data = node.next.data;
        prev = node;
        node = node.next;
     }

     prev.next = null;
  }
```



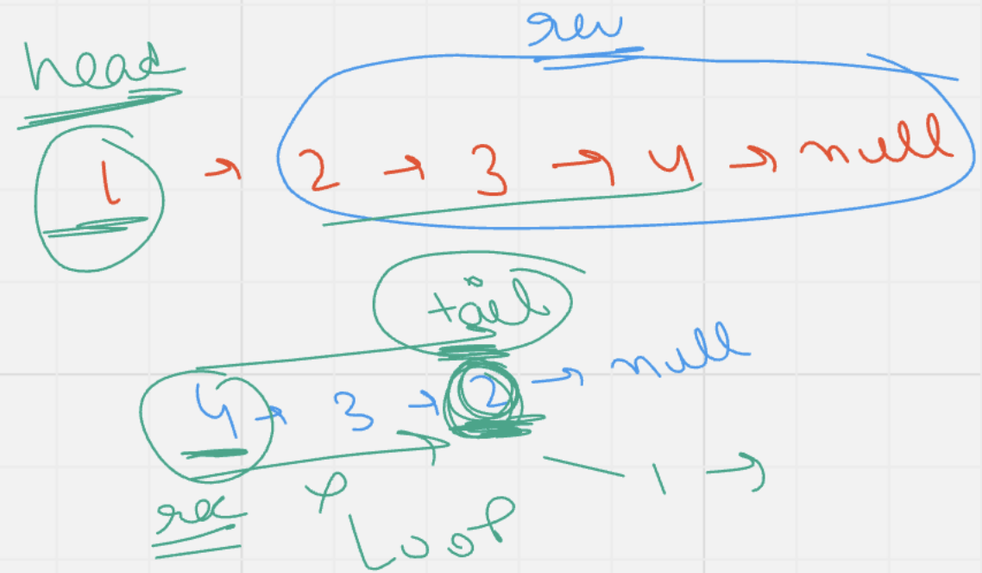


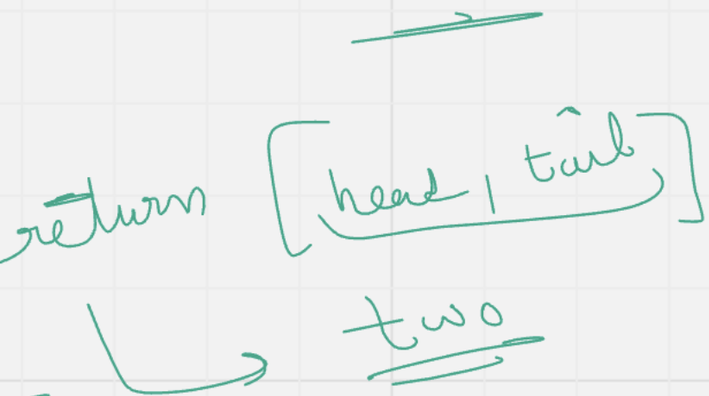

6 2
1 2 3 4 5 6
nth last = len - n + 1
6 - 2 + 1
5
from start
2
1
1 → 2 → 3 → 4 → 5 → 6 → null
0
1 → 2 → 3 → 4 → 6 → null
HW

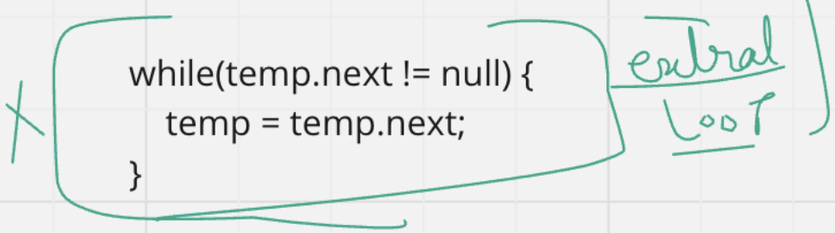

Reverse a LL
way 1
Rec N²
way 2
Rec N
way 3
Iter N
head
1 → 2 → 3 → 4 → 5 → null
recursion
rec (head → next)
5 → 4 → 3 → 2 → null
1
revhead
temp
5 → 4 → 3 → 2 → null
1
temp
head
correct
temp.next = head
head.next = null

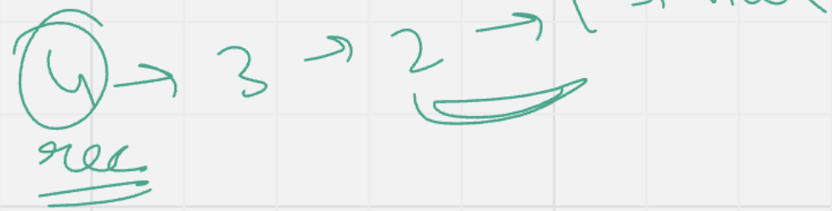

head
1 → 2 → 3 → 4 → 5
rev
5 → 4 → 3 → 2 → null
1 → null
rev
5 → 4 → 3 → null
5 → 4 → 3 → 2 → null
head
2 → 3 → 4 → 5
rev
5 → 4 → null
5 → 4 → 3 → 2 → null
head
3 → 4 → 5
rev
5 → 4 → 3 → 2 → null
head
4 → 5
5 → 4 → null
5 → null
Base Case

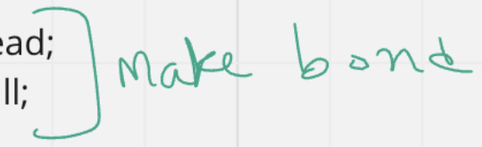
```
static Node revUsingRec(Node head) {

    if(head.next == null) {
        return head;
    }


    Node recHead = revUsingRec(head.next);
    Node temp = recHead;

    while(temp.next != null) {
        temp = temp.next;
    }

    temp.next = head;
    head.next = null;

    return recHead;

}
```

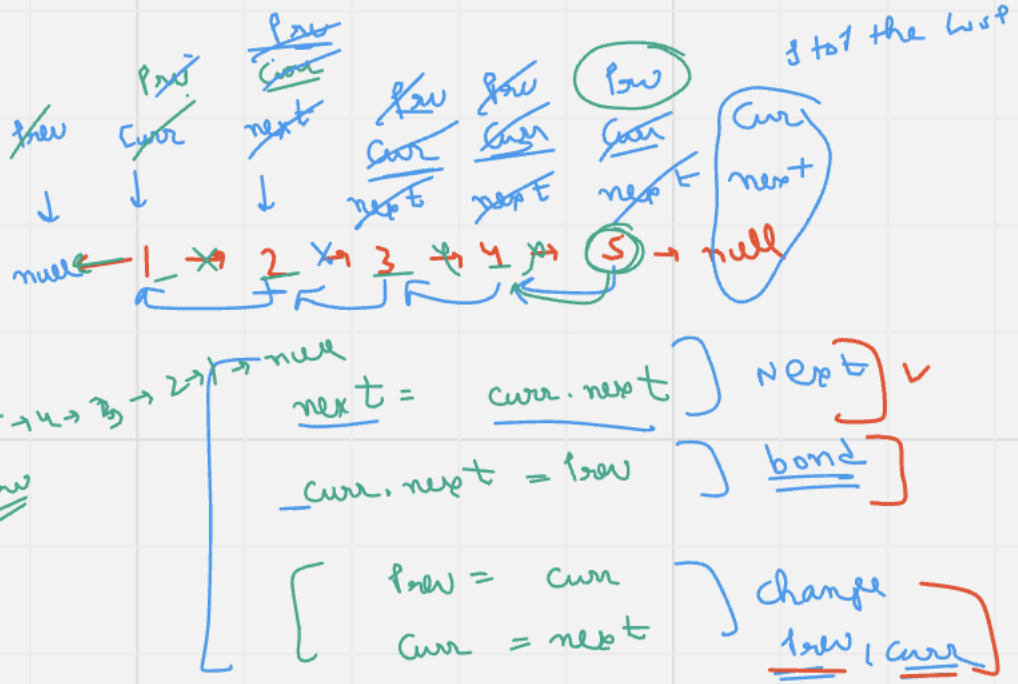


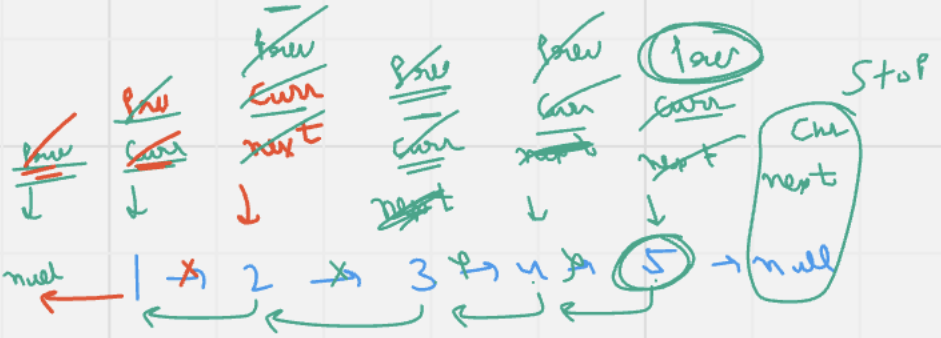


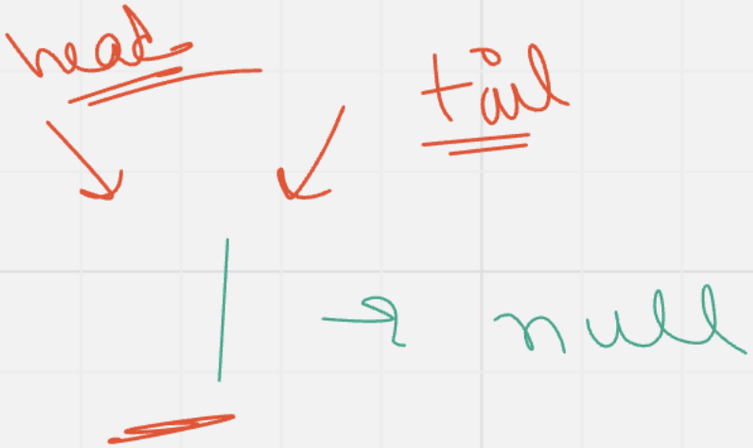


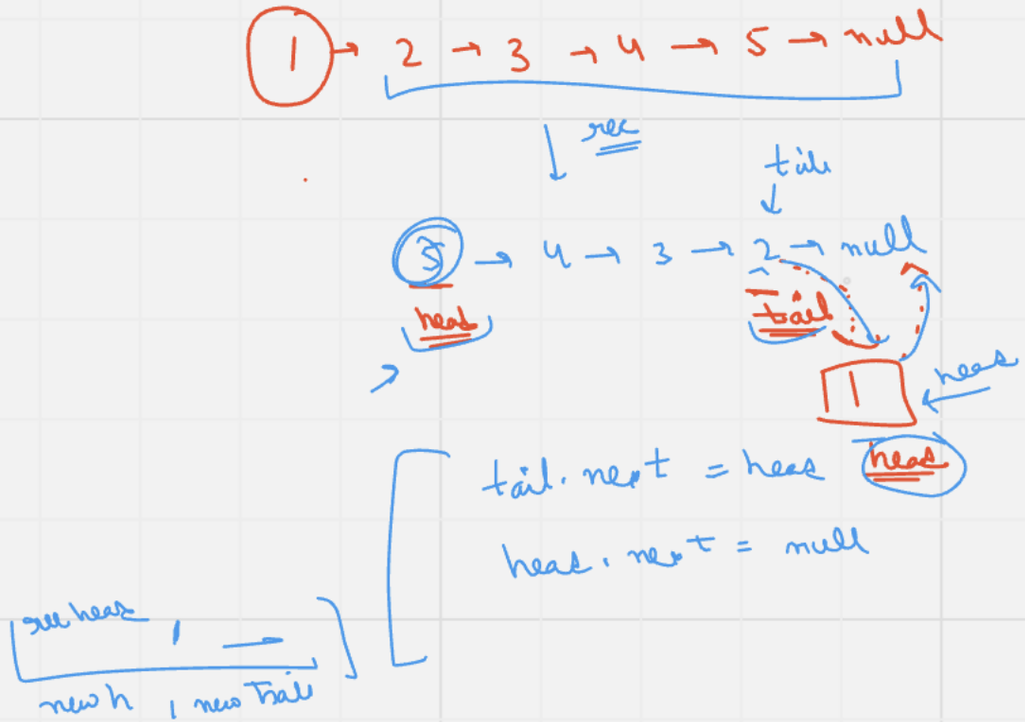


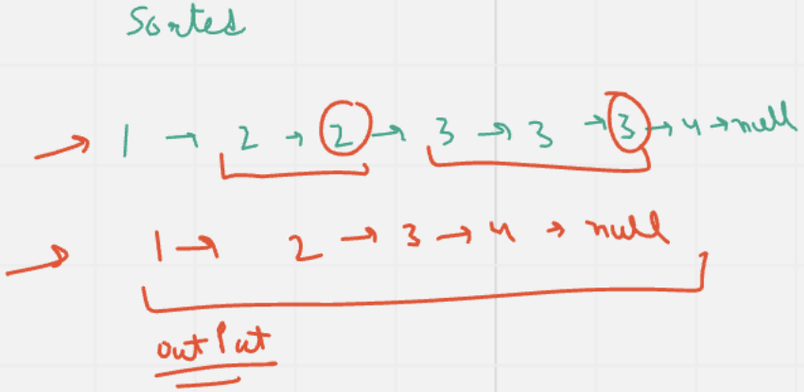


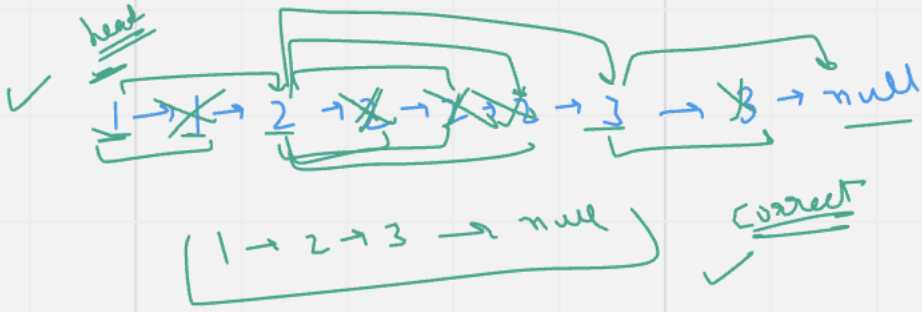

1 to the last
prev
curr
next
null
1
2
3
4
5
null
curr
next
null
5 → 4 → 3 → 2 → 1 → null
prev
next = curr.next
curr.next = prev
bond
prev = curr
curr = next
change
prev, curr

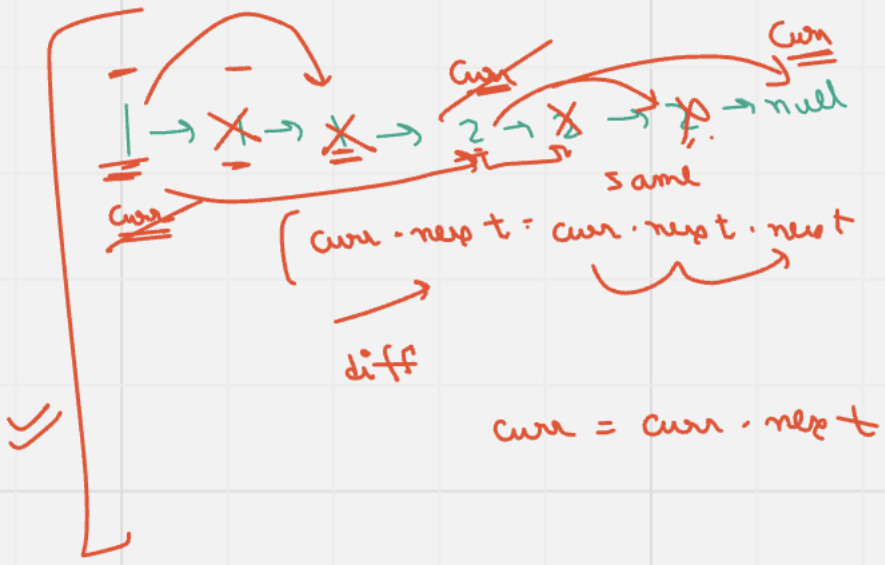

Stop
prev
curr
next
null
1
2
3
4
5
null
curr
next

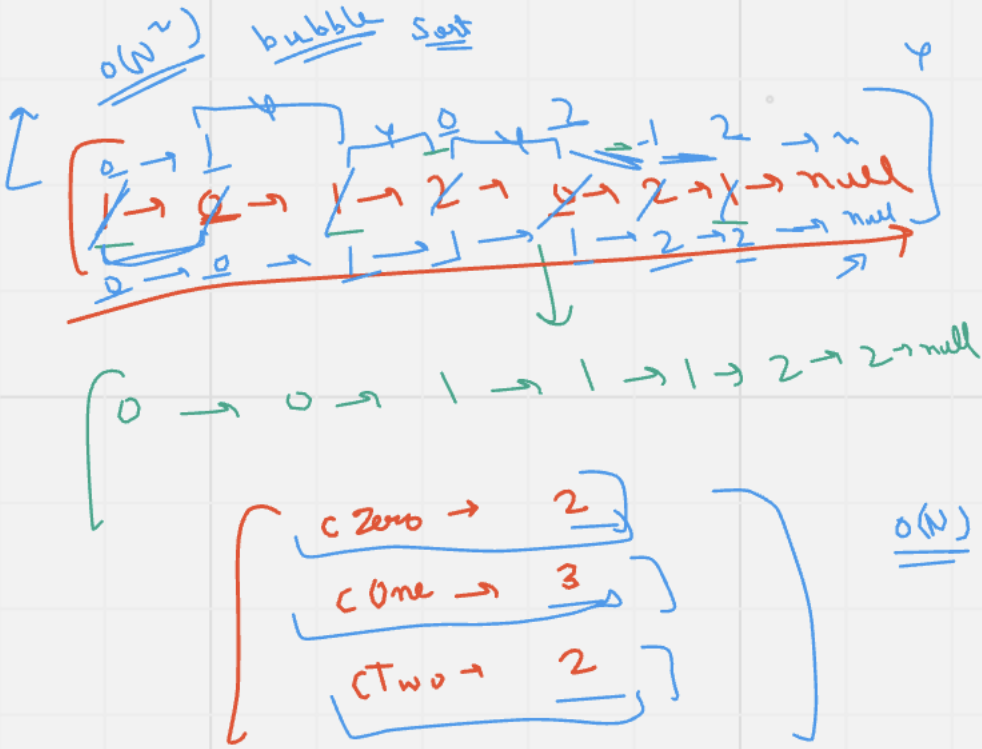

head
tail
null

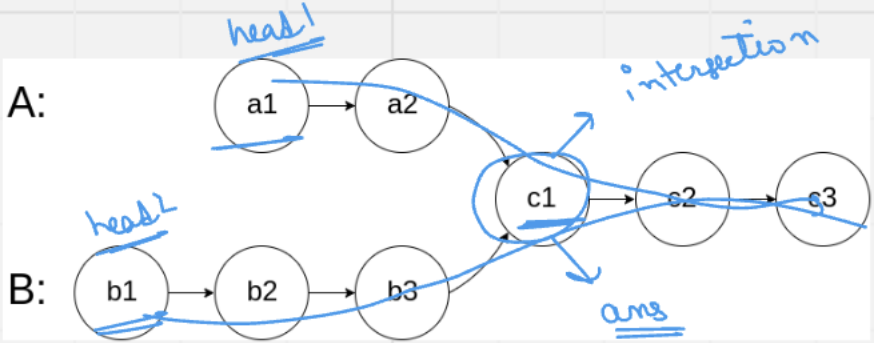

1 → 2 → 3 → 4 → 5 → null
rev
tail
5 → 4 → 3 → 2 → null
head
tail
head
1
head
tail.next = head
head.next = null
rev head
new h
new Tail

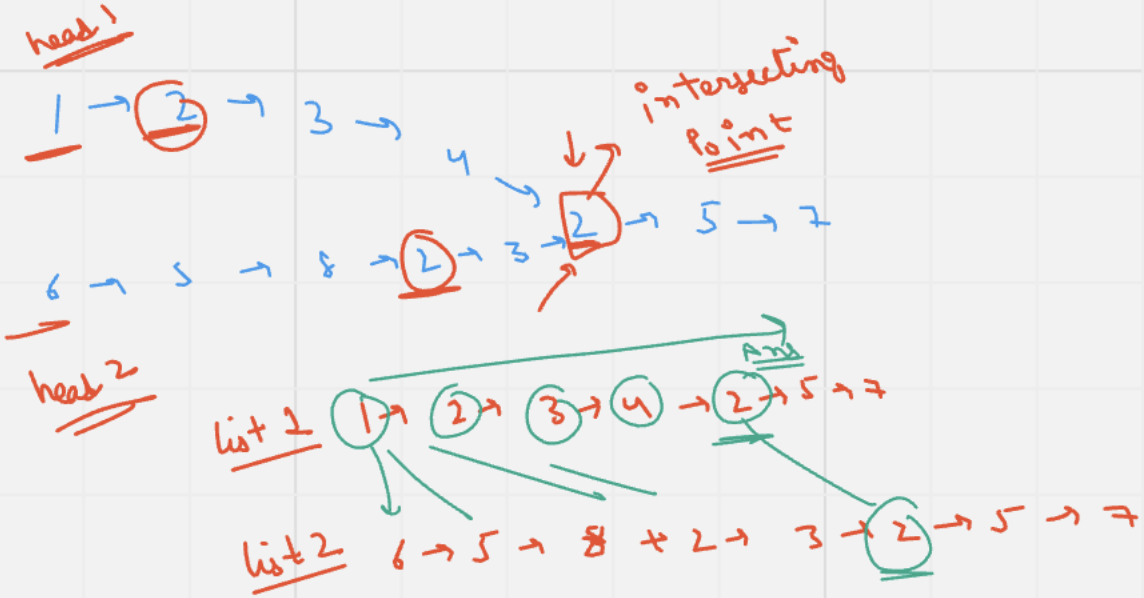

Sorted
1 → 2 → 2 → 3 → 3 → 3 → 4 → null
1 → 2 → 3 → 4 → null
output

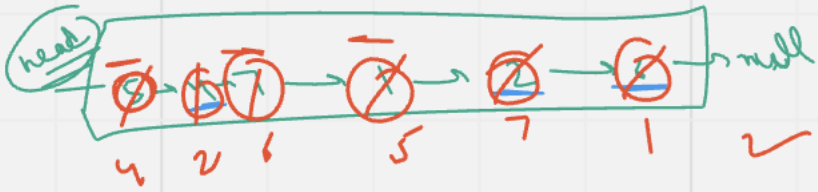

head
1 → 2 → 3 → null
1 → 2 → 3 → null
correct


curr
curr
curr
null
same
curr.next = curr.next.next
diff
curr = curr.next

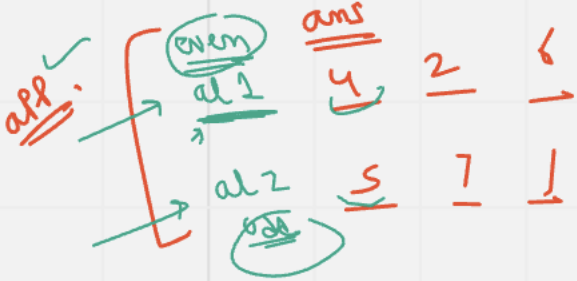

O(N²)
bubble sort
0 → 0 → 1 → 1 → 1 → 2 → 2 → null
cZero → 2
cOne → 3
cTwo → 2
O(N)

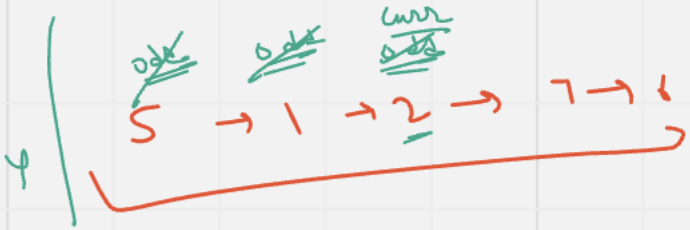

head1
A:
a1
a2
intersection
c1
c2
c3
head2
B:
b1
b2
b3
ans

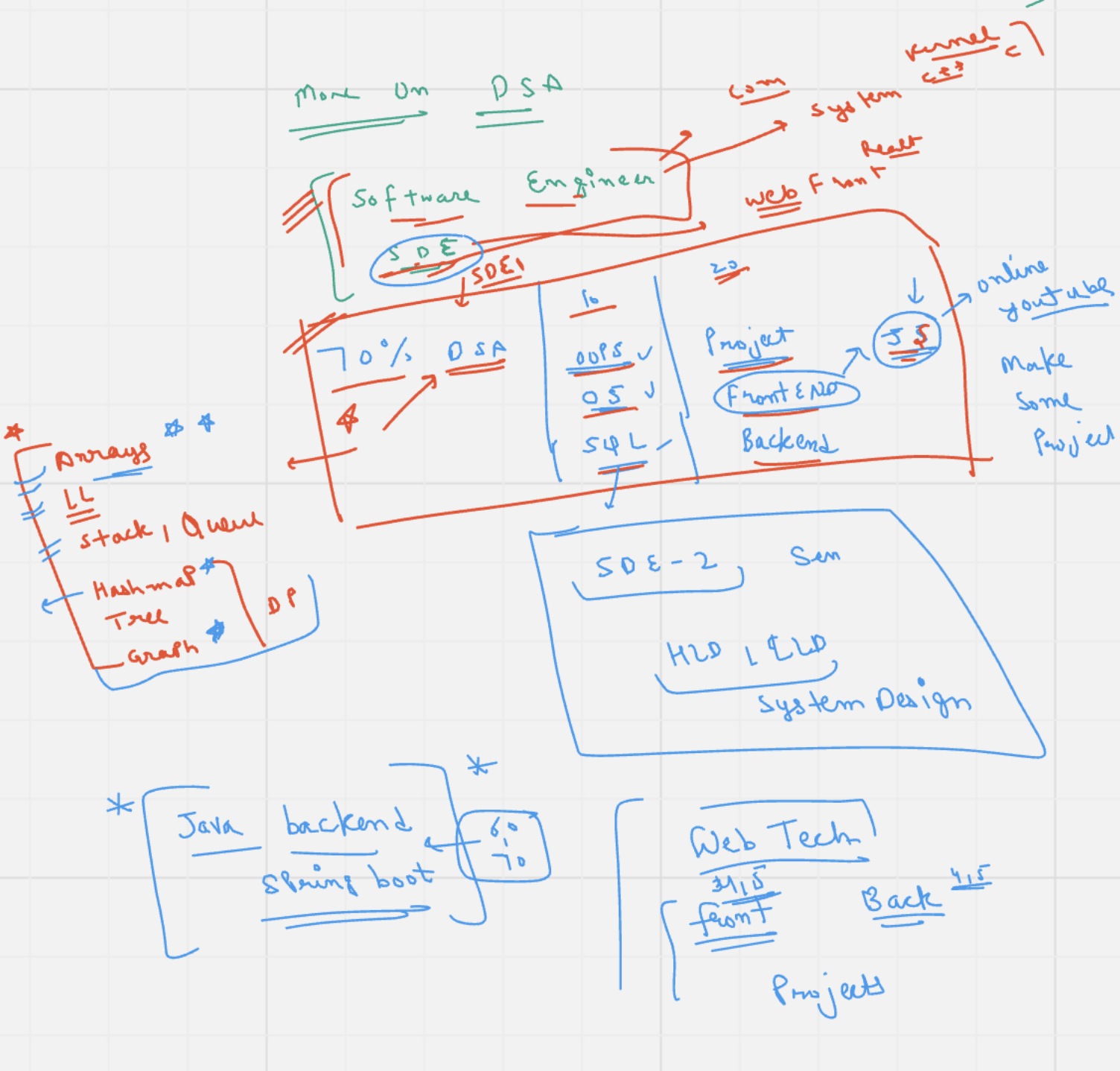

head1
intersecting point
head2
list 1
list 2
ans


head
null
4
2
1
5
7
1
2


even
odd
4 → 2 → 1 → 5 → 7 → 1 → null


ans
even
odd
4
2
1
5
7
1


odd
odd
curr
5 → 1 → 2 → 7 → 1


Kernel
C++
C
More on DSA
Com
System
Software Engineer
Web Front
SDE
SDE1
2+
1+
70% DSA
OOPS
OS
SQL
Project
FrontEND
Backend
JS
online youtube
Make some Project
Arrays
LL
stack / Queue
Hashmap
Tree
Graph
DP
SDE-2
Sem
HLD / LLD
System Design
Java backend
Spring boot
6+ / 7+
Web Tech
front
Back
Projects