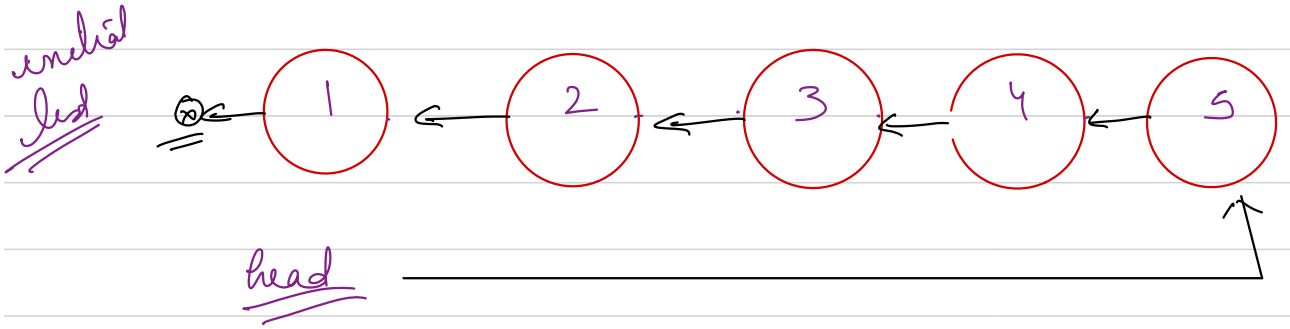
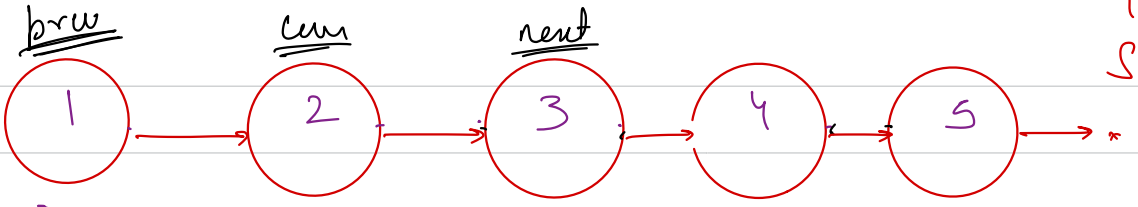
Q:) Given a linked list of integer values in a node, Reverse the linked list, by only manipulating the next pointers. (Donot create a new LL).



Try to solve both iteratively and recursively

Reverse Pointers Iterative



TC → O(n)
SC → O(1)

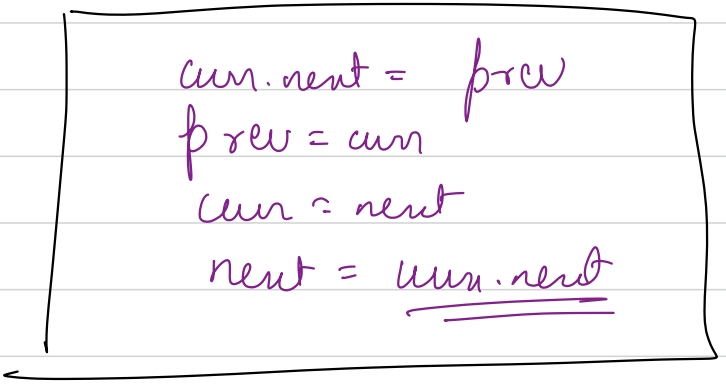
head

At last (5) should become the head.

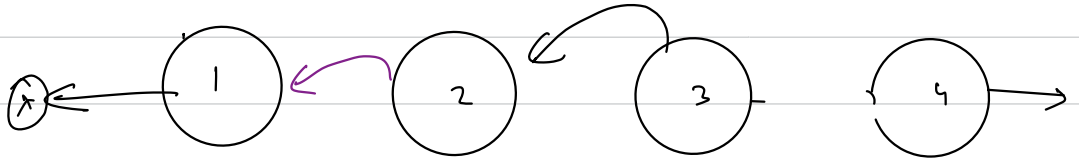
→ Before manipulating the ptr of any node, we should store the rest of the LL somewhere

→ Any node after the process should point to its prev node

prev , curr , next



core logic



prev = null	1	2	3
curr = 1	2	3	4
next = 2	3	4	null
initial state	1st iteration	2nd iteration	iteration stops here

Reverse Pointer Recursive

↳ Base Case

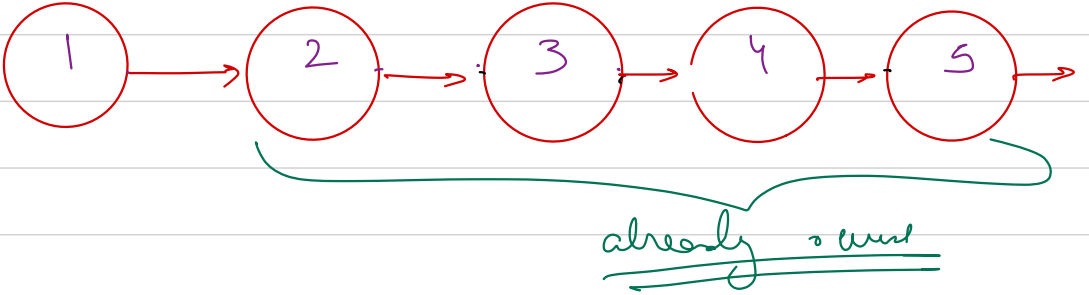
↳ Self work

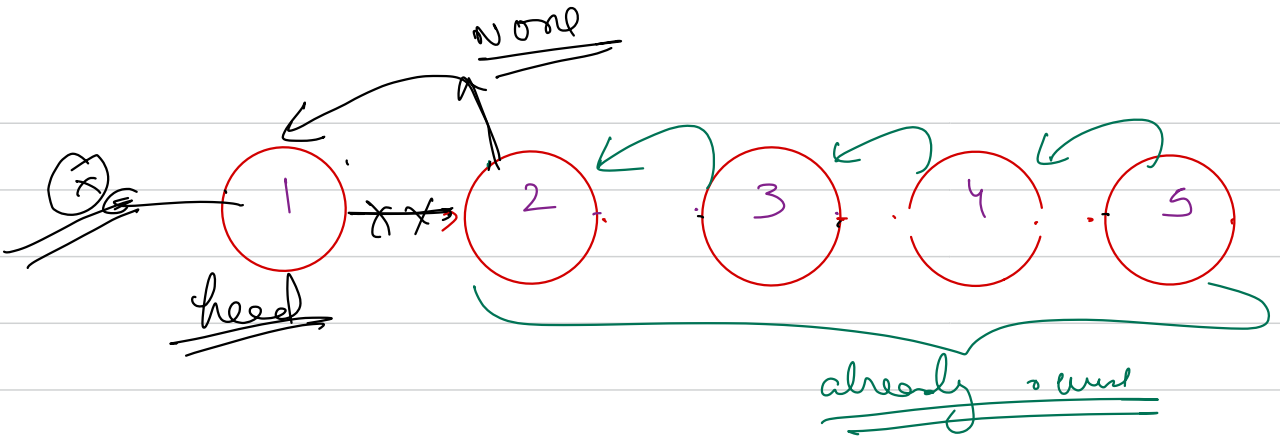
↳ Recursive Intuition

TC → O(n)

SC → O(n) ↓ due to recursive call stack

Base Case → Empty LL or LL of size 1





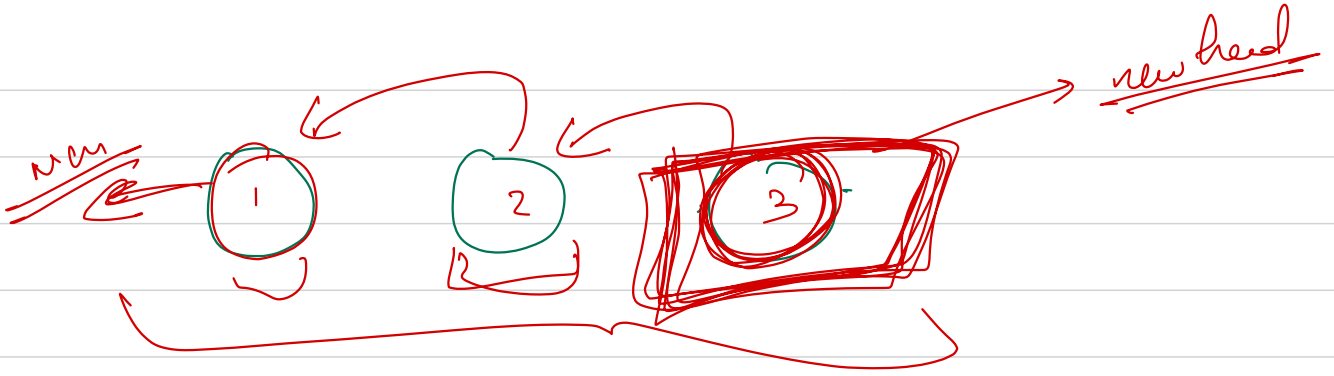
$f(head) = f(head.next)$ → recursive intuition

↓

reverse the LL

↳ self work as → $head.next.next = head$

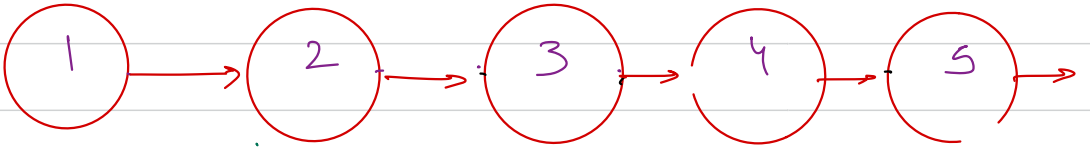
$head.next = None$


new head
1
2
3

Reverse LL Data Iterative

You have to reverse a ll without manipulating pointers & only replacing data

(Modify the original LL)



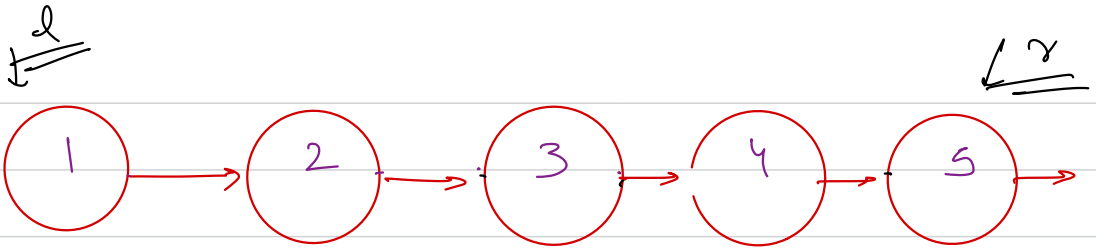
TC → $O(n^2)$

SC → $O(1)$

i = 0

n - i - 1

Reverse Data Recursively



We have a benefit with recursion (palendromic LL)

when you should swp ??

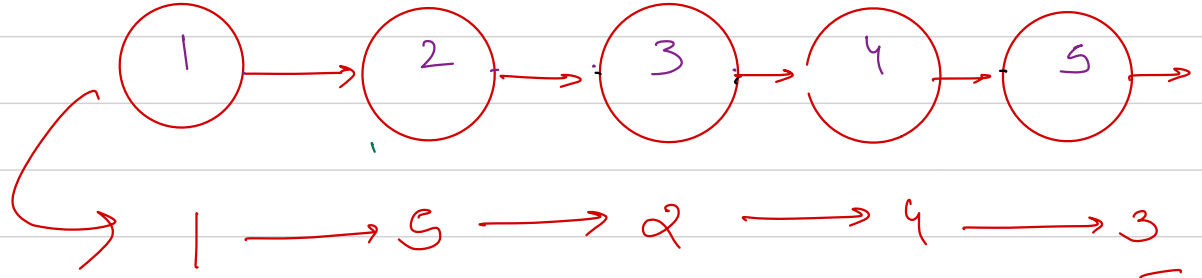
odd → l == r ← swp here

even → l.next == r stop

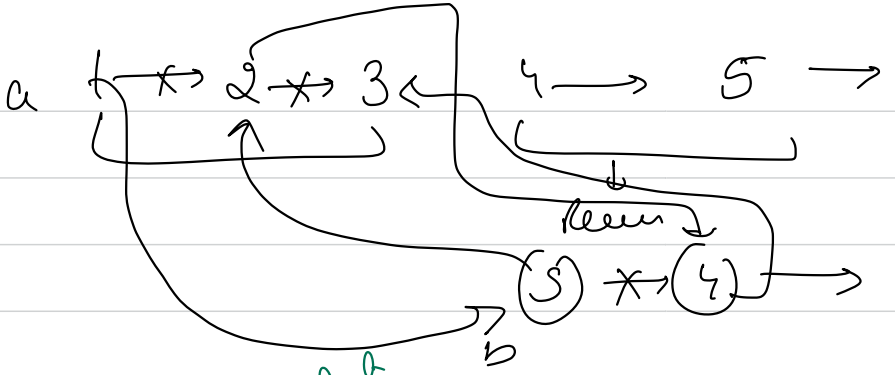
Q: Given a linked list, rearrange the LL in place from $L_0 \to L_1 \to L_2 \to L_3 \cdots\cdots L_{n-1} \to L_n$ it transforms to

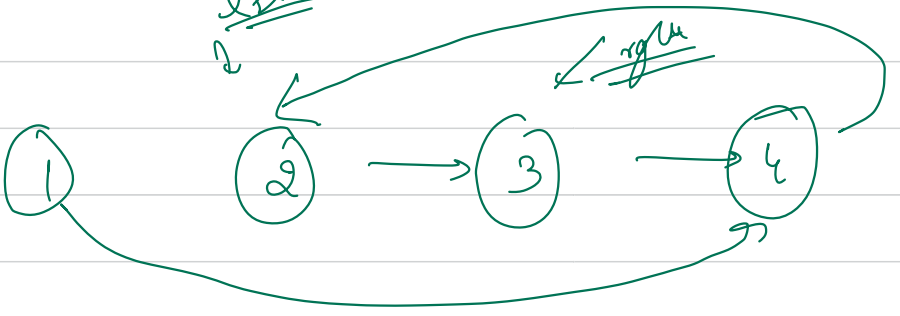
Hint → Palindromic LL

$L_0 \to L_n \to L_1 \to L_{n-1} \to L_2 \to L_{n-2} \cdots\cdots$



we can't do data swaps





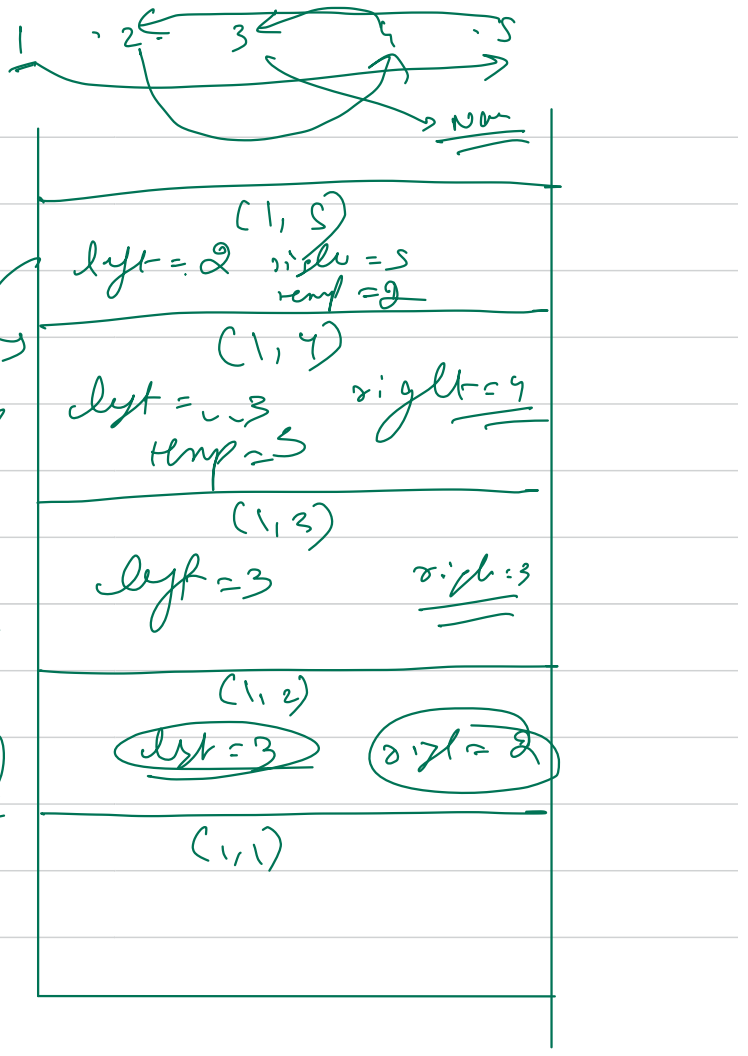
$L_0 \rightarrow L_n \longrightarrow L_1 \longrightarrow L_{n-1}$

```python
def fold_helper(left, right):
    if right == None:
        return left

    left = fold_helper(left, right.next)
    if left == None:
        return left

    if left != right and left.next != right:
        temp = left.next
        left.next = right
        right.next = temp
        left = temp
    else:
        right.next = None

    return left
```



Q. You've a ll, with every node having a next ptr & bottom ptr. flatten the ll. all are LL is sorted

5 → 10 → 19 → 28

5 → 7 → 8 → 30

10 → 20

19 → 22 → 50

28 → 35 → 40 → 45

In-place

Merge 2 sorted LL

↳ 5 → 7 → 8 → 10 → 19 → 20 → 22 → 28 → 30 → 35 → 40 → 45 → 50

→ flatt → head.next

5 → 10 → 19 → 20 → 22 → 28 → 35 → 40 → 45 → 50

↓

7

↓

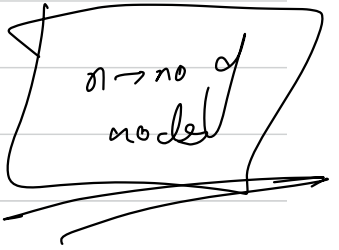
8

↓

30

O(n)

O(n)



(1) → head.next

(2) → head.bottom