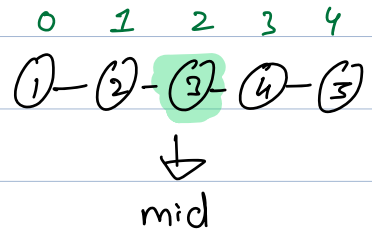
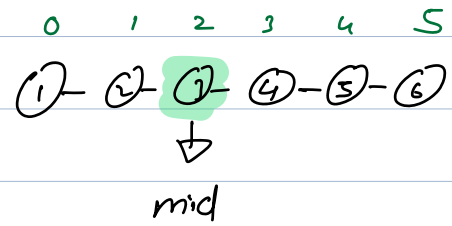
Q1 Given a linked list, find the middle element of linked list.

Ex1



Ex2



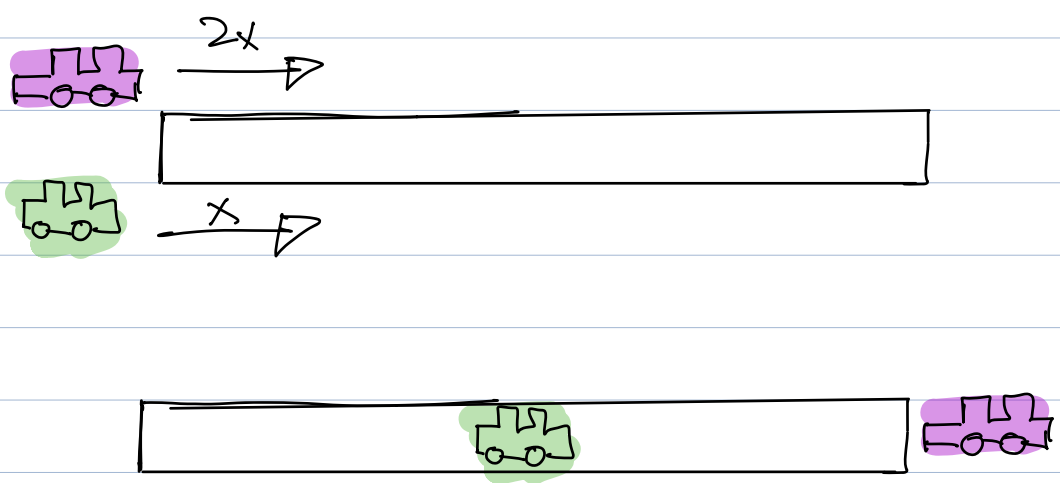
Approach 1: 1) Find length

2) Iterate to $\frac{\text{length}}{2}$.

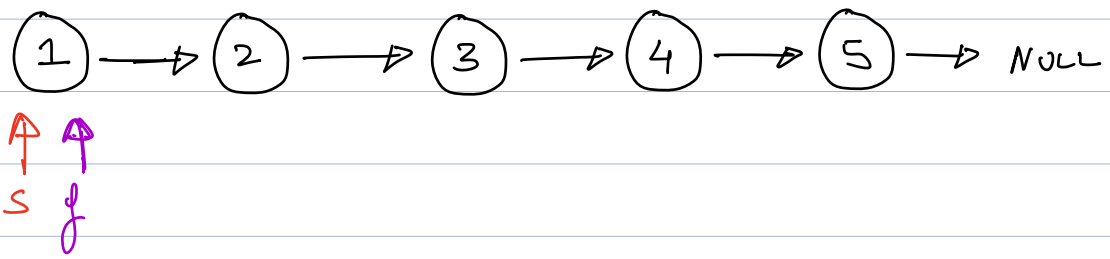
Tc : $O(n)$ [Total iterations $\Rightarrow n + n/2$ $\Rightarrow 3n/2$]

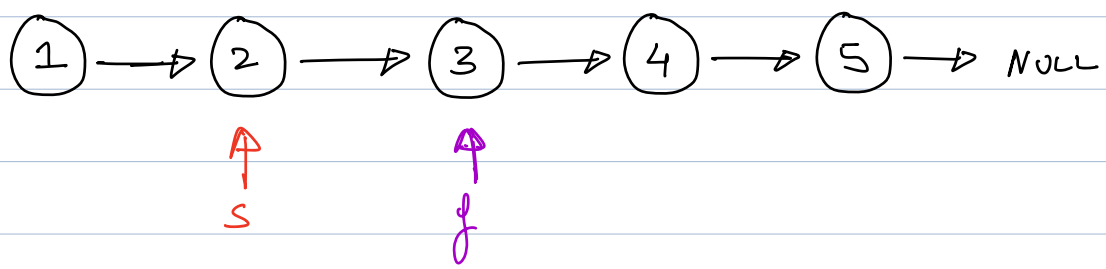
Sc : $O(1)$

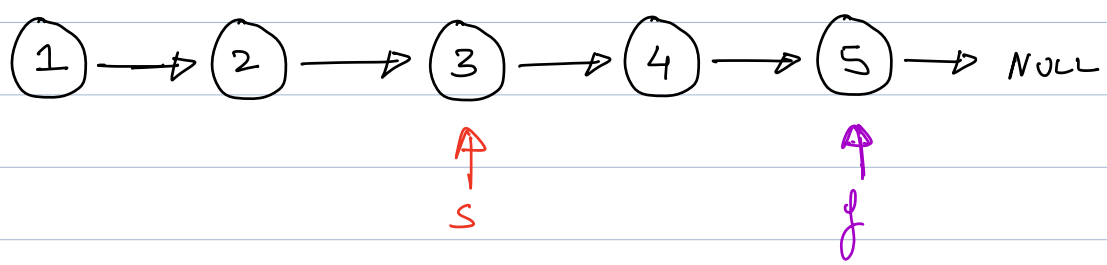
Approach 2



ODD LENGTH

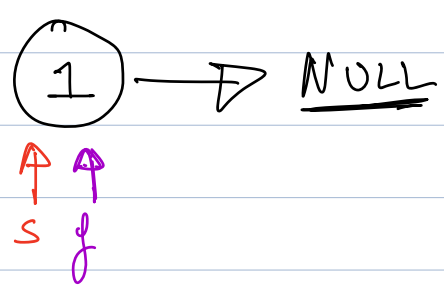




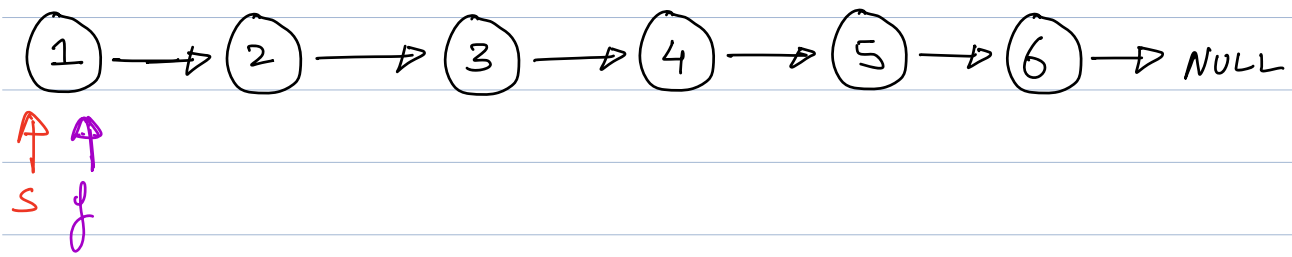


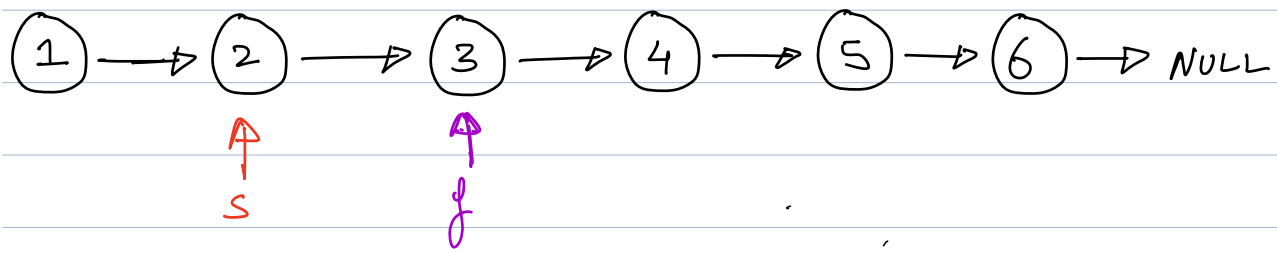
Terminating Condition : f.next == null

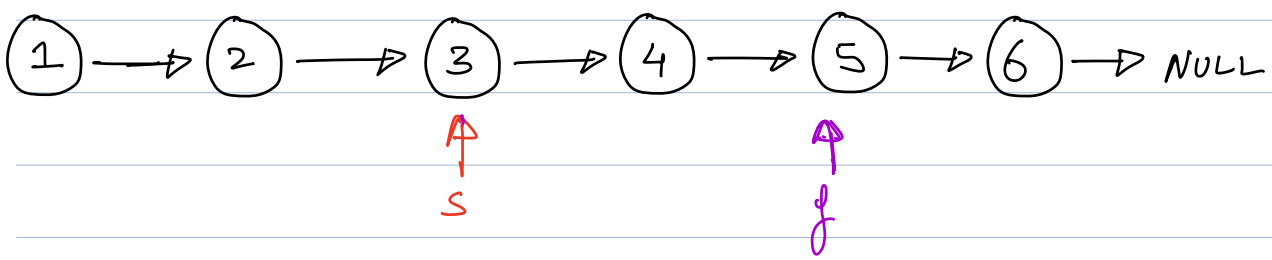
#



EVEN LENGTH

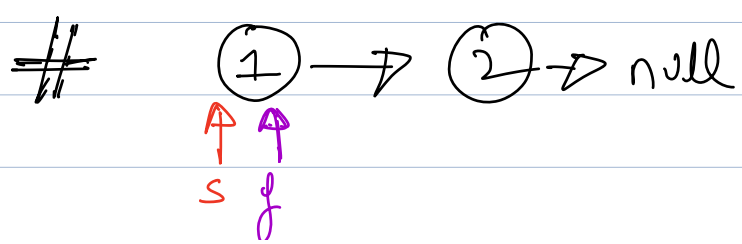






Terminating Condition : f.next.next == null

Edge Cases : 1) Head == null



Pseudo Code !

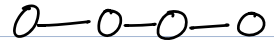
```
if (head == null)
        return null;

Node slow = head , fast = head.

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

return slow;
```

TC: $O(n)$ ⇒ n iterations

SC: $O(1)$

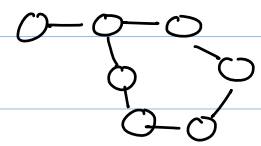
Q2 Find if a cycle exists in a linked list.

Ex1



$\Rightarrow$ FALSE

Ex2



$\Rightarrow$ TRUE

Approach 1 : $n \leq 10^4$ $\Rightarrow$ Cool but that's it

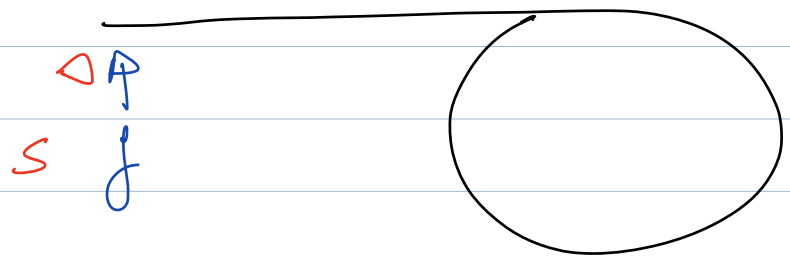
$\Downarrow$

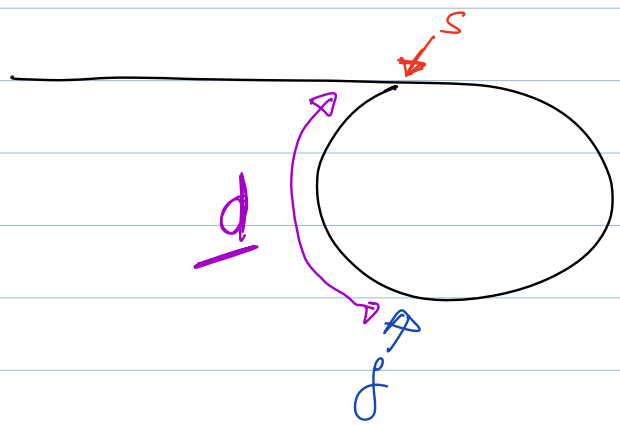
Traverse for $n = 10^4$ times, if you reach null $\Rightarrow$ no cycle
else $\Rightarrow$ cycle.

Approach 2 : Keep inserting in the set
If a node is encountered twice $\Rightarrow$ there is a loop

TC : $O(n)$
SC : $O(n)$

Approach 3 : Space Optimization.





$t \Rightarrow 0$

Distance betwen

s & f $\Rightarrow$ d.

$t = 1$

Distance betwen

s & f $\Rightarrow$ $d-1$.

$t = d$

Distance betwen

s & f $\Rightarrow$ 0

$s \Rightarrow 1$

$f \Rightarrow 3$

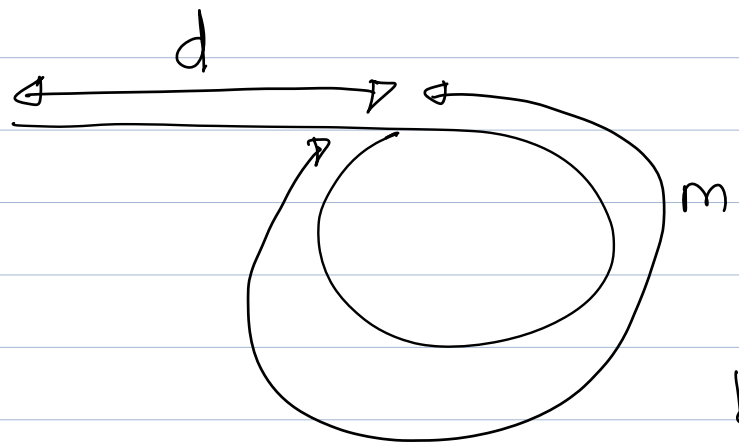
$s \Rightarrow 2$

$f \Rightarrow 3$

```
if (head == null)
        return null;

Node slow = head, fast = head.

while (fast.next != null && fast.next.next != null) {
    fast = fast.next.next;
    slow = slow.next.

    if (fast == slow) {
        return true;
    }

}

return false;
```

TC : O(n)

SC : O(1)



$d + m \Rightarrow n$

slow: $d+m$

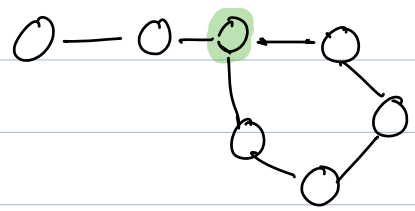
$\Rightarrow n$

max distance b/w slow & fast

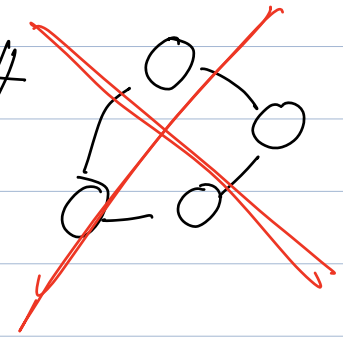
$\Rightarrow m$

Q3 If a loop exist, find the starting point of the loop.

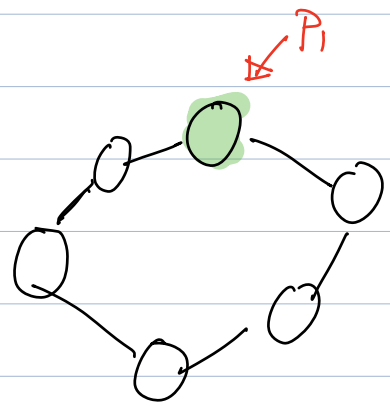
Ex1



#



#



length ⇒ 6

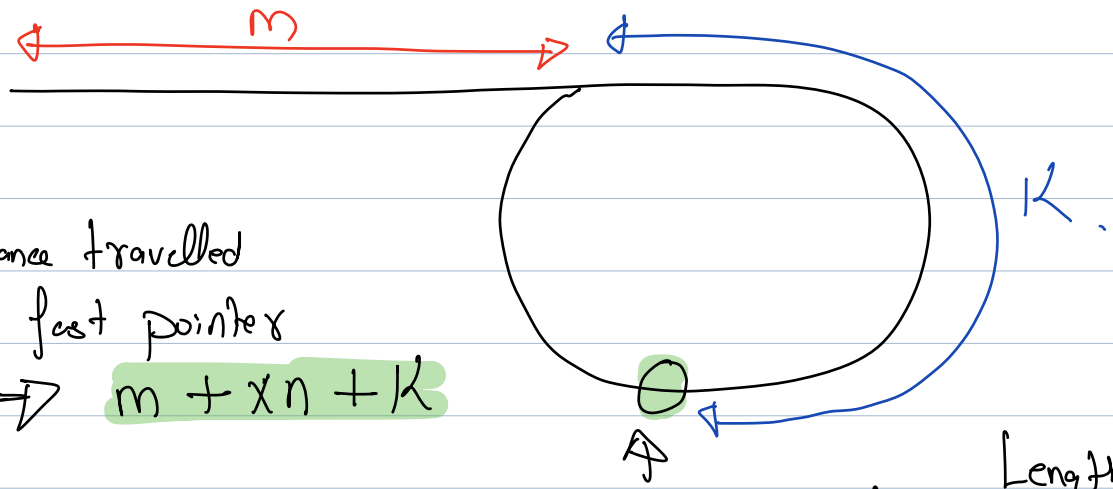
P_1 travel 6 steps

" " 12 steps.

P_1 travel x steps.

Length of the cycle ⇒ n.

If x is a multiple of n, P_1 reaches the point where it started from.



Distance travelled by fast pointer $\Rightarrow m + xn + K$

Distance travelled by slow pointer $\Rightarrow m + yn + K$

meeting point of fast and slow

Length of the loop $\Rightarrow n$.

$$\text{Dis (fast)} \Rightarrow 2\,\text{Dis (slow)}$$

$$m + xn + K \Rightarrow 2(m + yn + K)$$

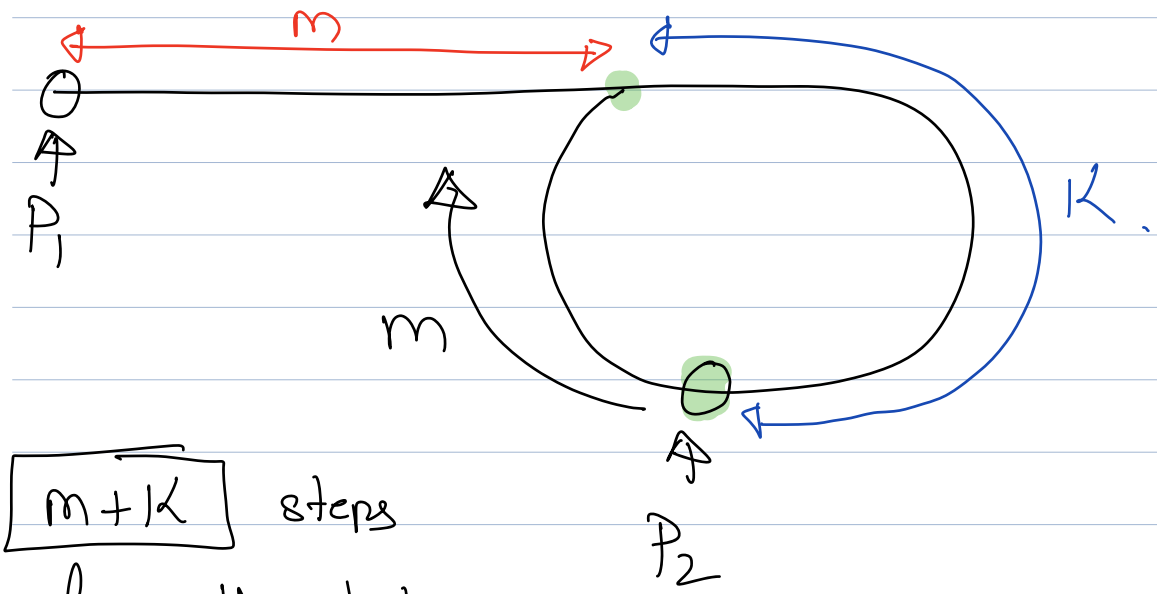
$$m + xn + K \Rightarrow 2m + 2yn + 2K$$

$$xn - 2yn \Rightarrow m + K$$

$$m + K \Rightarrow n(x - 2y)$$

$$m + K \Rightarrow n\{z\}$$

$m + K$ is a multiple of n.



$\boxed{m+K}$ steps from the starting point of the loop.

Pseudo Code

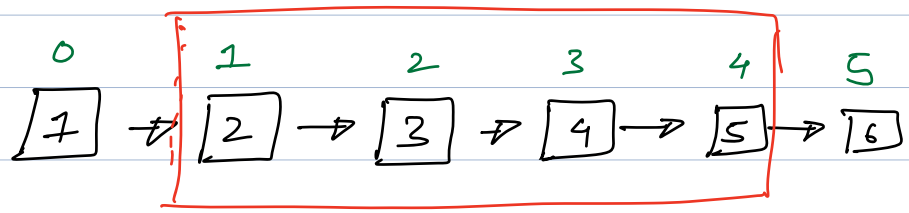
1) Find meeting point.

```
if (slow == fast) {

    Node p1 => head;
    Node p2 => slow;

    while (P1 != P2) {
        p1 = p1.next;
        p2 = p2.next;
    }

    return p1;
}
else {

    return null;
}
```

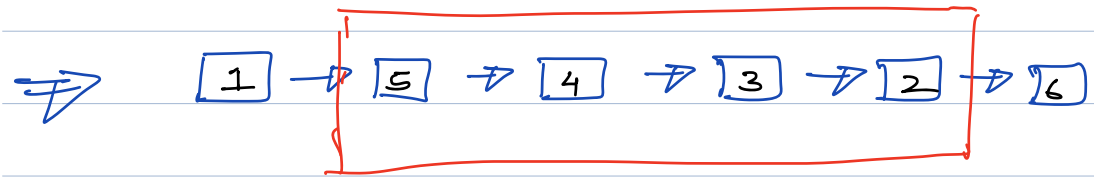
TC : O(n)
SC : O(1)

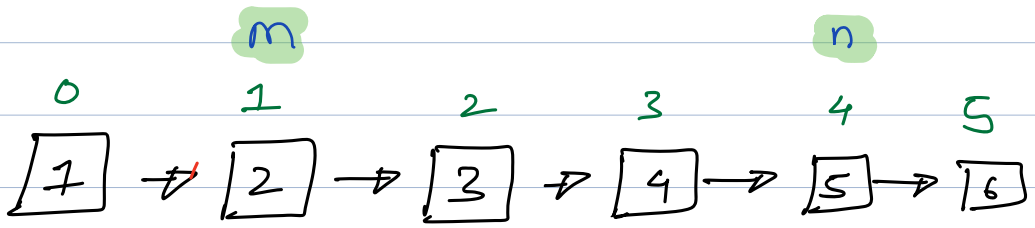
Q4 Reverse a part of a linked list

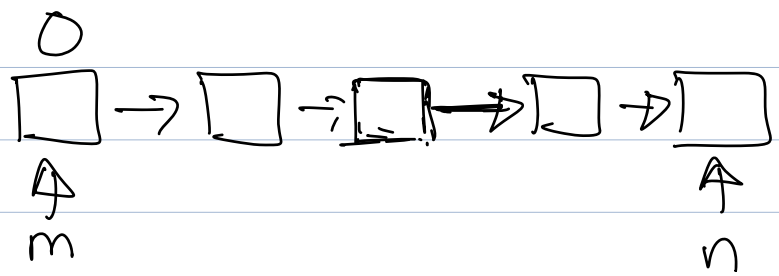
Ex1



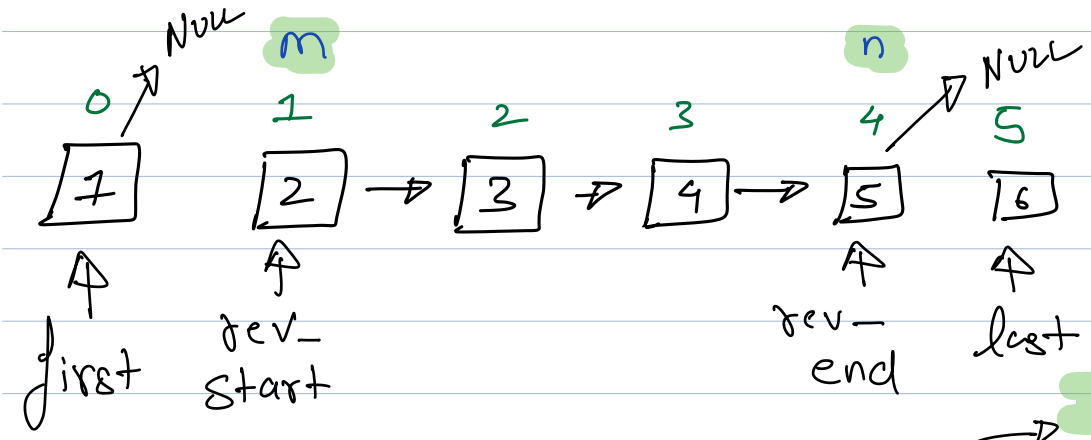
$m = 1$, $n = 4$







1) Extract



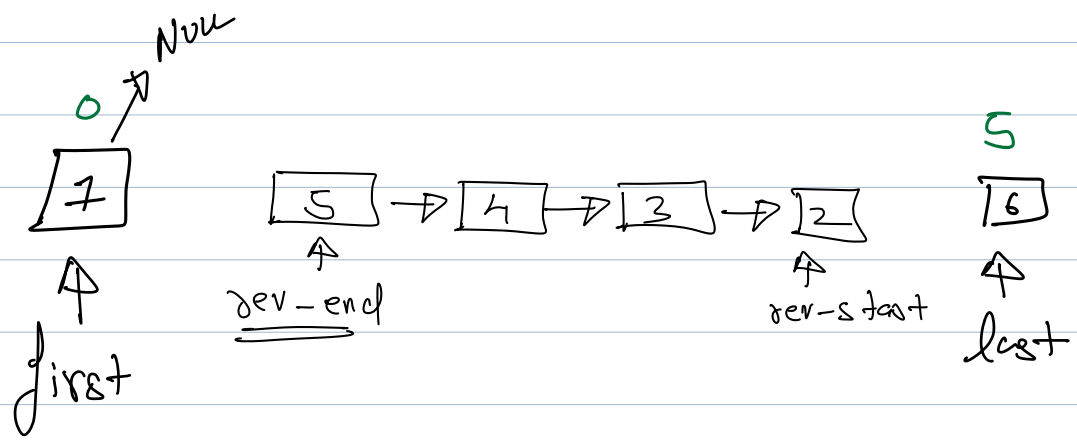
→ if (first != null)

first.next = null

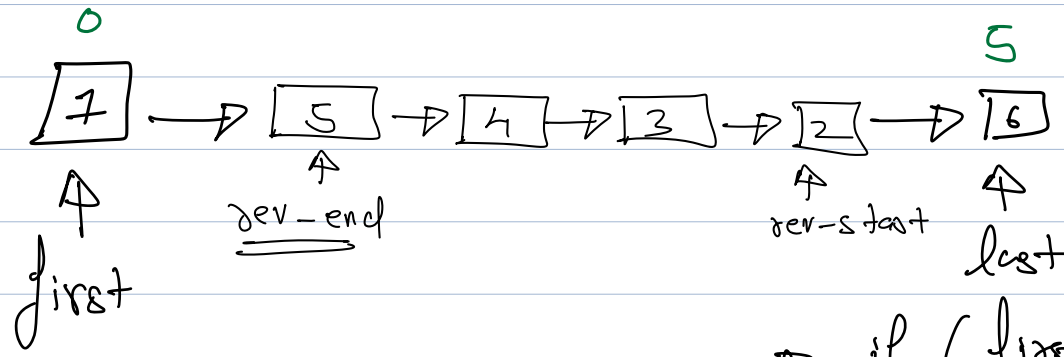
rev_end.next = null

2) Reverse

reverse (rev_start)



3) Join back.




```
if (first != null)
    first.next = rev_end
    rev_start.next = last
```

Q5 Rearrange linked list !

Ex1

$[1] \to [2] \to [3] \to [4] \to [5]$

$\Rightarrow [1] \to [5] \to [2] \to [4] \to [3]$

Ex2

$[1] \to [2] \to [3] \to [4] \to [5] \to [6]$

Approach

1) Find mid.

$[1] \to [2] \to [3] \to [4] \to [5] \to [6]$

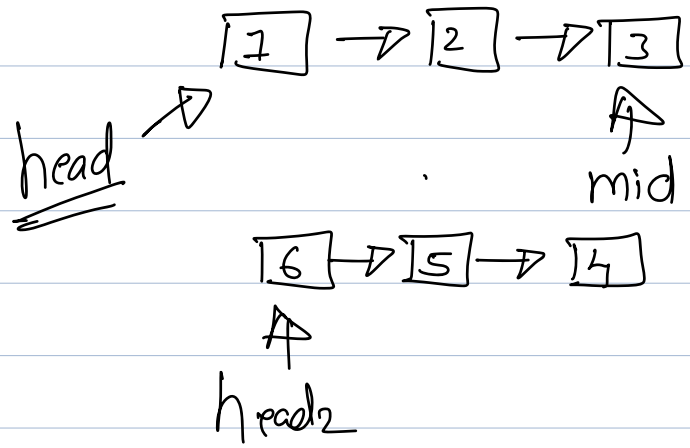
mid.

2) Break from mid.

$[1] \to [2] \to [3]$ $\quad$ $[4] \to [5] \to [6]$

mid. $\quad$ head2

3) Reverse the second list



4) Join the list