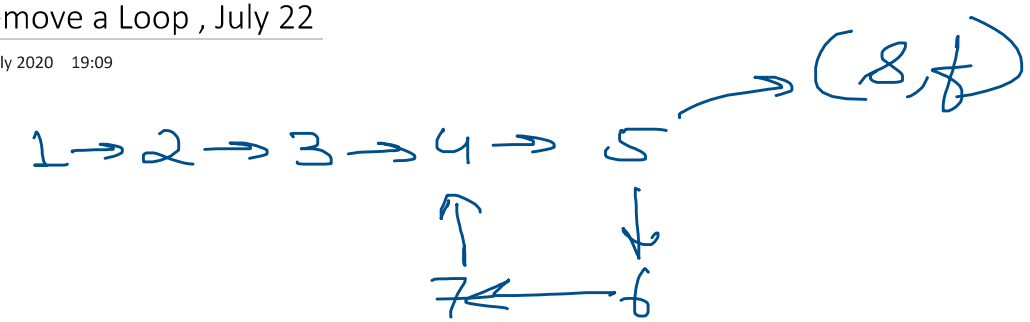
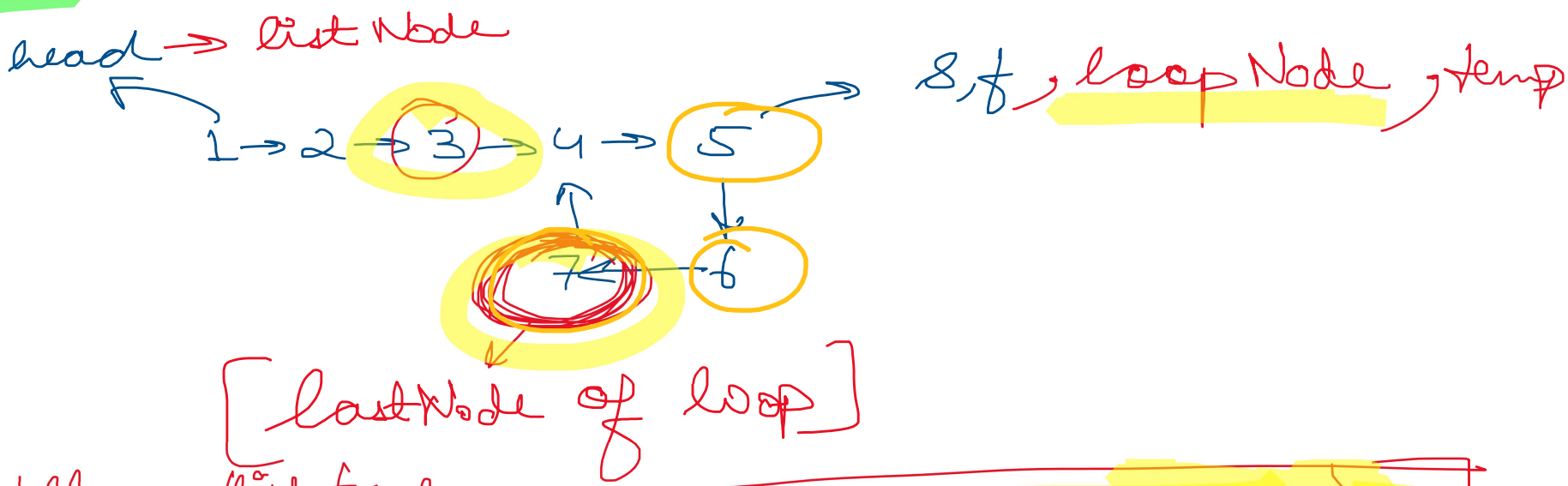
Remove a Loop , July 22

22 July 2020 19:09



Approach 1



When this true

(loopNode.next == ListNode.next)

(last Node of loop)

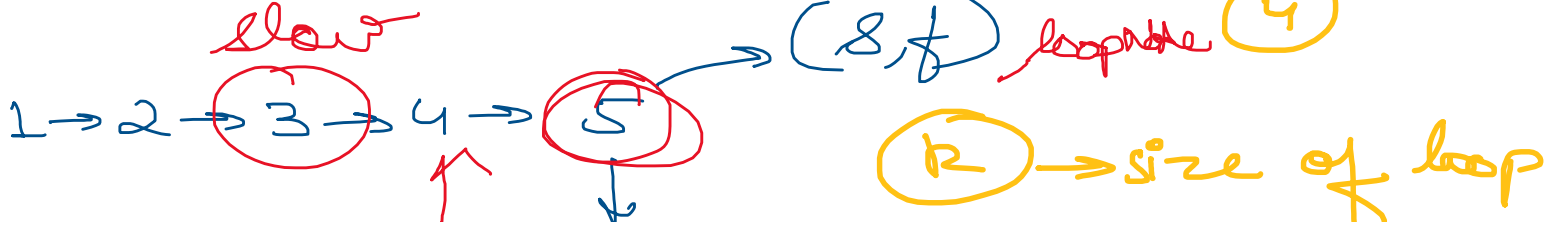
loopNode.next = null

```
void removeLoop(Node listNode, Node loopNode){
    Node temp = loopNode;
    while(true){  //To increment listNode
        while(loopNode.next!=listNode.next && loopNode!=temp) //To increment loopNode
        {
            loopNode = loopNode.next;
        }
        if(loopNode.next==listNode.next) //if required node found{
            //loopNode.next = null;
            //return;
            break;
        }
        listNode = listNode.next;
    }
    loopNode.next = null;
}
```

Complete 1 circle

Required Node found

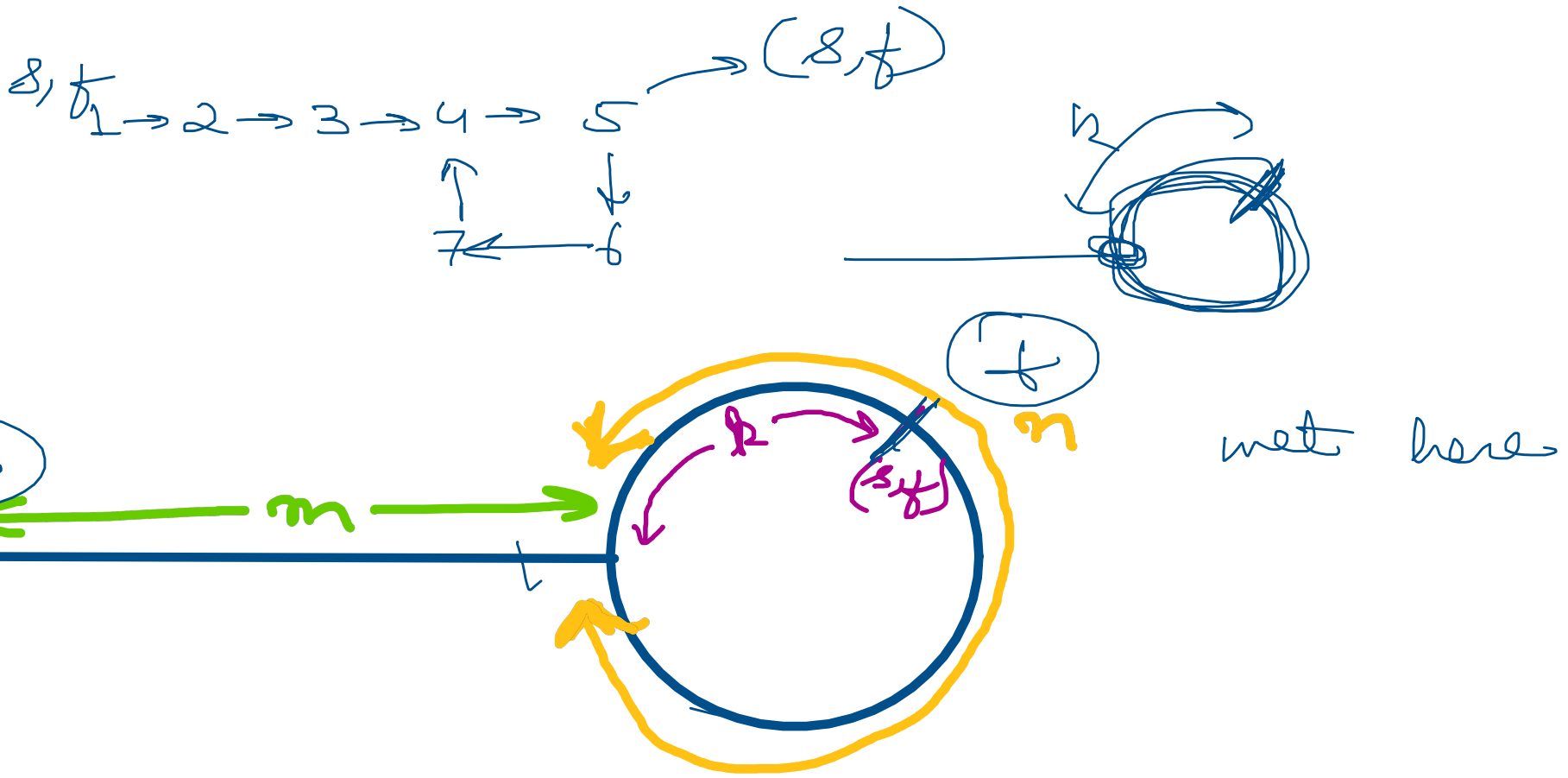
Approach-2



Step 1 = find size of loop

```
void removeLoop(Node head, Node loopNode){
    Node temp = loopNode;
    int count = 1;
    while(loopNode.next!=temp)
    {
        loopNode = loopNode.next;
        count++;
    } //count=size of loop
    Node slow = head;
    Node fast = head;
    for(int i= 0;i<count;i++)
    {
        fast=fast.next;
    }
    while(slow.next!=fast.next){
        slow = slow.next;
        fast = fast.next;
    }
    //while(fast.next!=slow)
    //{
        //fast = fast.next;
    //}
    fast.next = null;
}
```

Approach-3 Most efficient



Distance covered by fast pointer

$= m + x \times n + k$

Distance covered by slow pointer

$= m + y \times n + k$

Distance by fast $= 2 \times$ Distance by slow

$m + (x \times n) + k = 2(m + y \times n + k)$

$m + xn + k = 2m + 2yn + 2k$

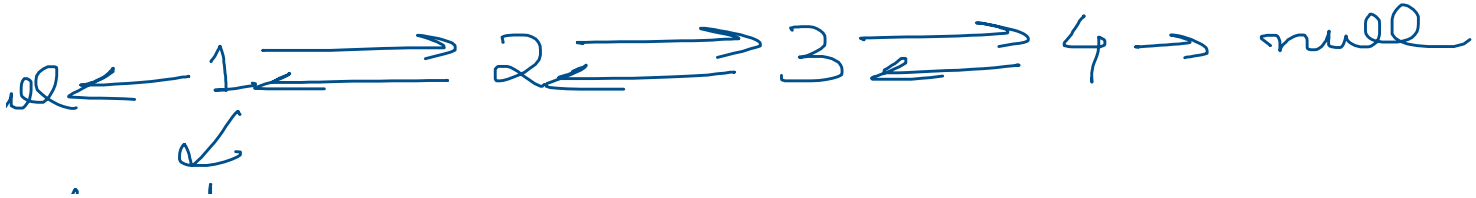
$n(x - 2y) = m + k$

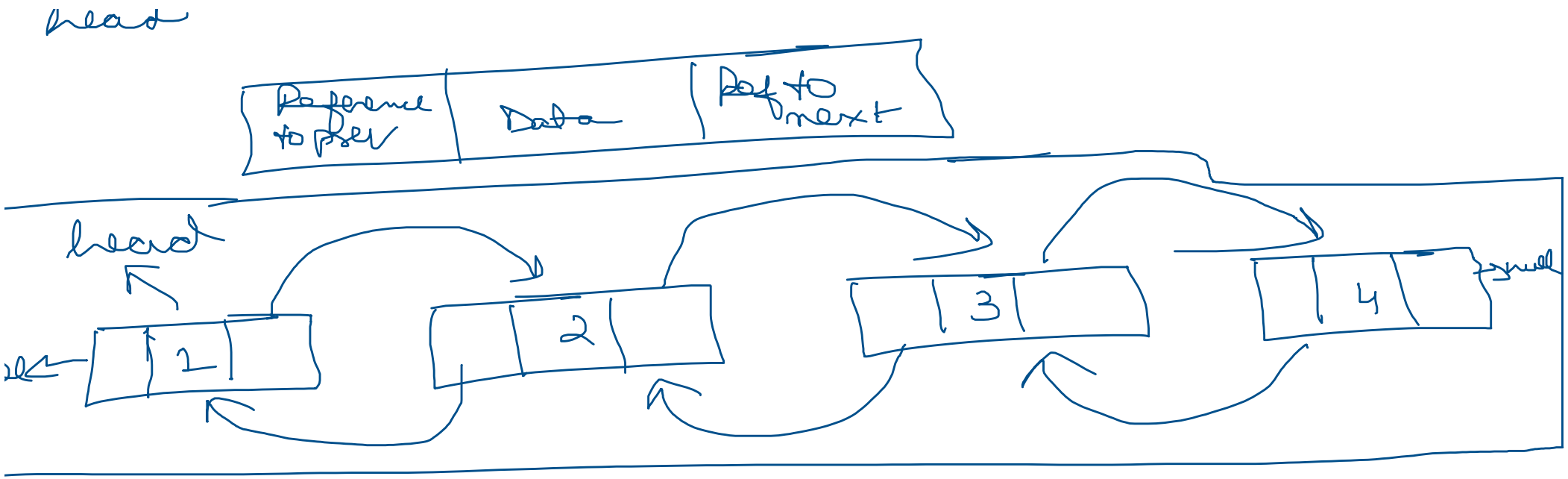
All are integers

$m+k$ is a multiple of n

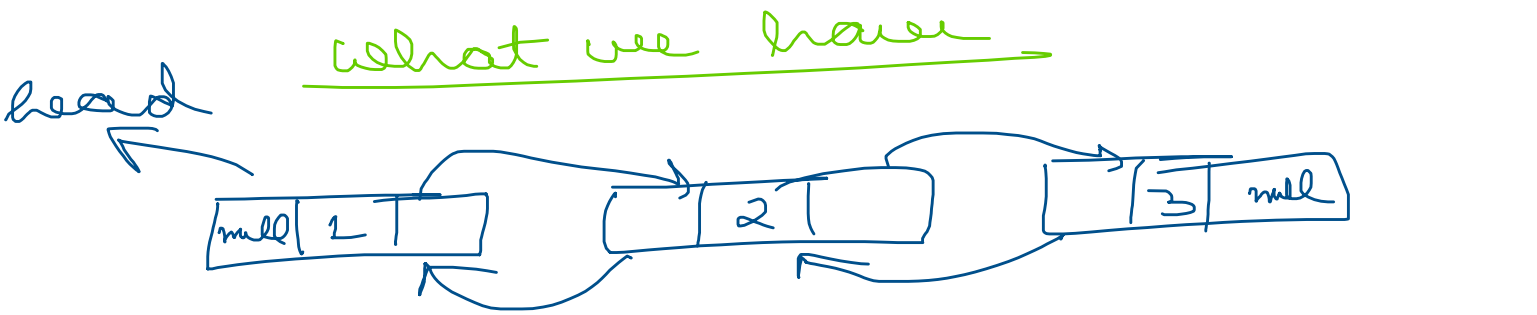
```
void detectAndRemoveLoop(Node head)
{
    if(head == null) return;
    Node slow = head;
    Node fast = head;
    while(fast!=null && fast.next!=null){
        if(slow==fast){
            break;
        }
        slow =slow.next;
        fast=fast.next;
    }
    //If loop exists
    if(slow==fast){
        slow = head;
        while(slow.next!=fast.next){
            slow=slow.next;
            fast=fast.next;
        }
        fast.next = null;
    }
}
```

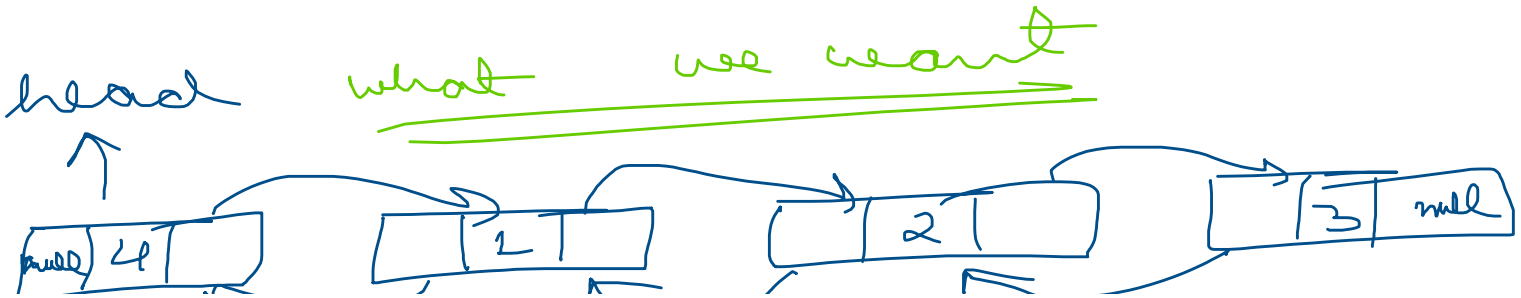
oubly Linked List



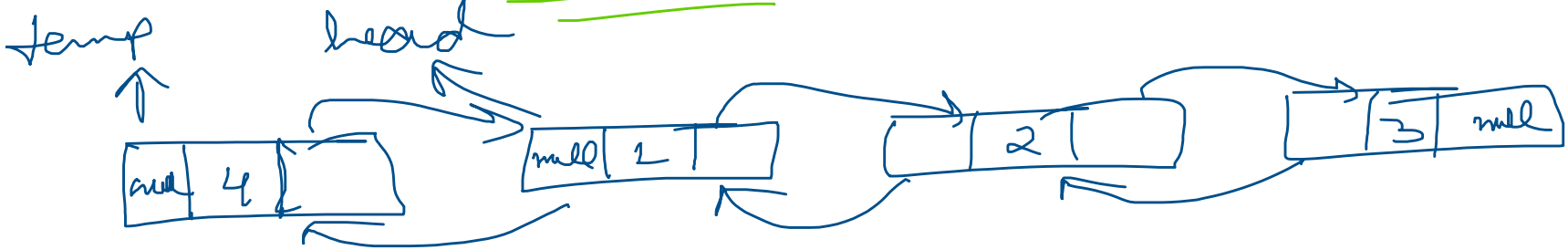


) InsertAt Head





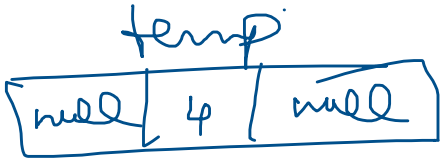
Steps




```
temp->next = head;
head->prev = temp;
return temp;
```

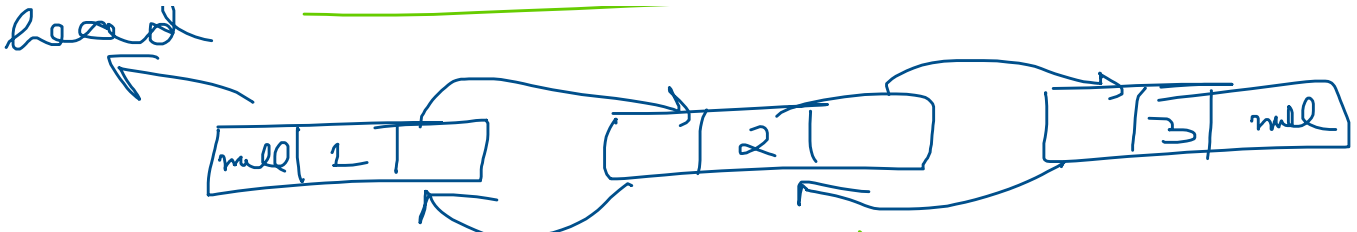
if LL empty

```
head = null
```

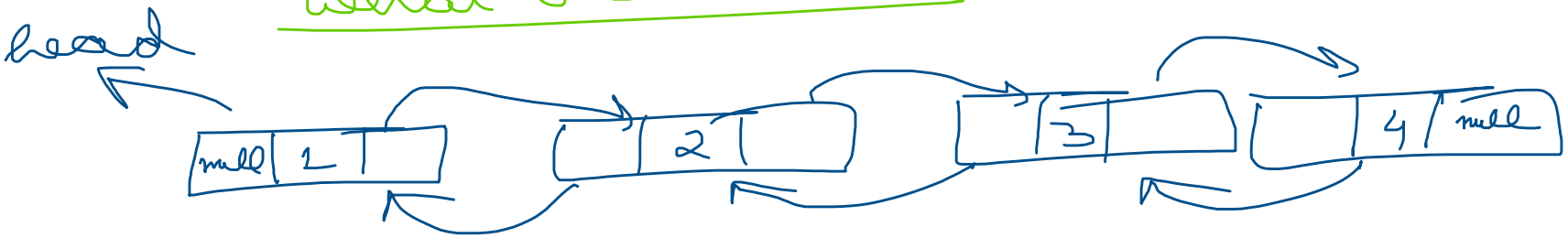


2) Insert At End

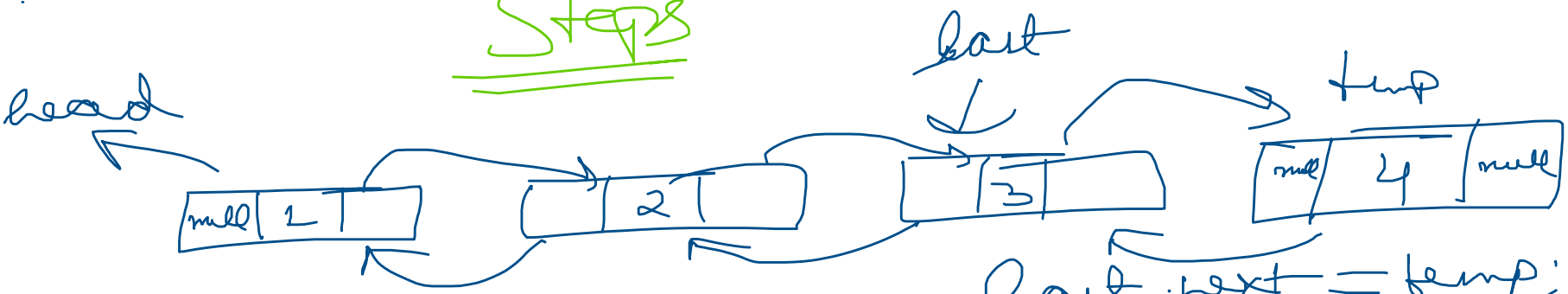
what we have



what we want



Steps



last.next = temp;

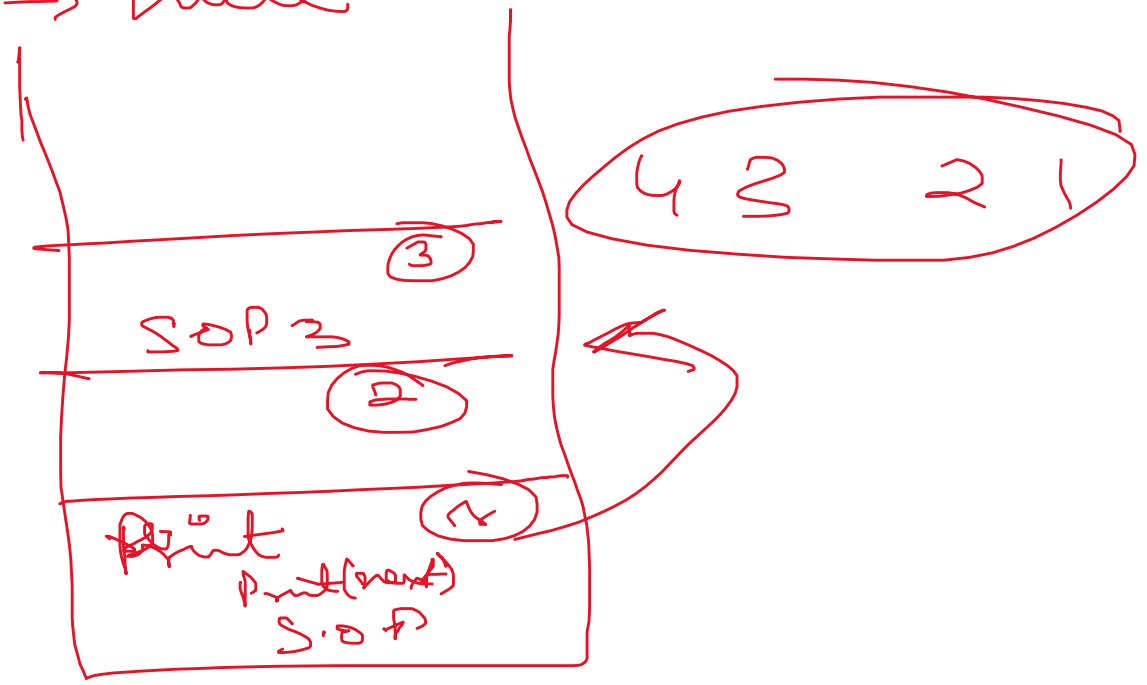
temp.prev = last;

If Empty

head = null

Overlapping using Recursion

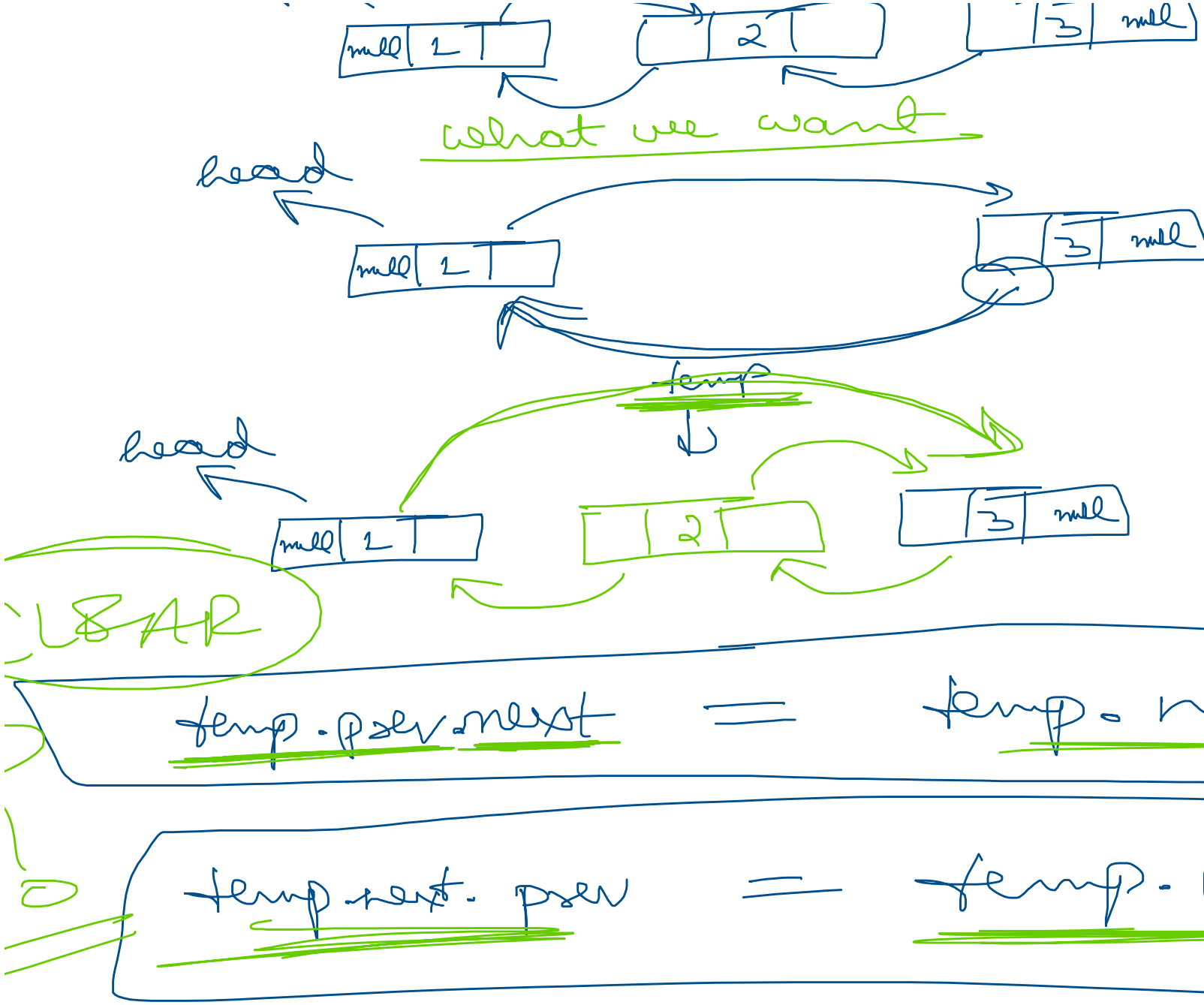
1 → 2 → 3 → 4 → null



Delete Node In B/w Doubly LL

what we have

head


null 1
2
3 null
what we want
head
null 1
3 null
temp
head
null 1
2
3 null
CLEAR
temp.prev.next = temp.next
temp.next.prev = temp.prev