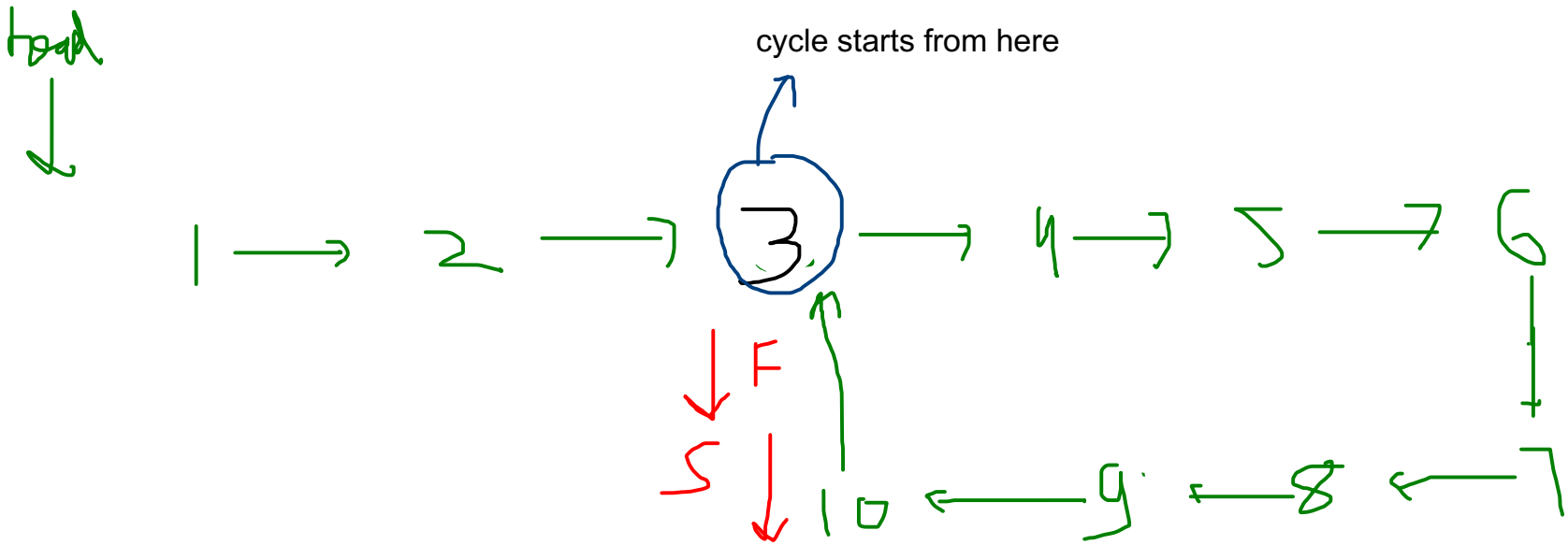
Detect Cycle point in LL

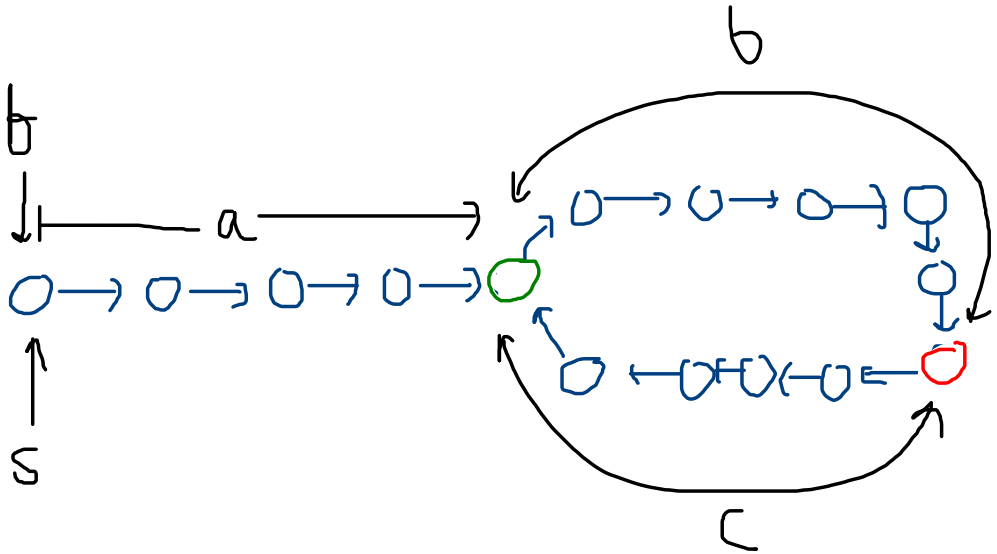


after meet of S and F

Reset slow to head

move both by one one

proof of cycle detection algo



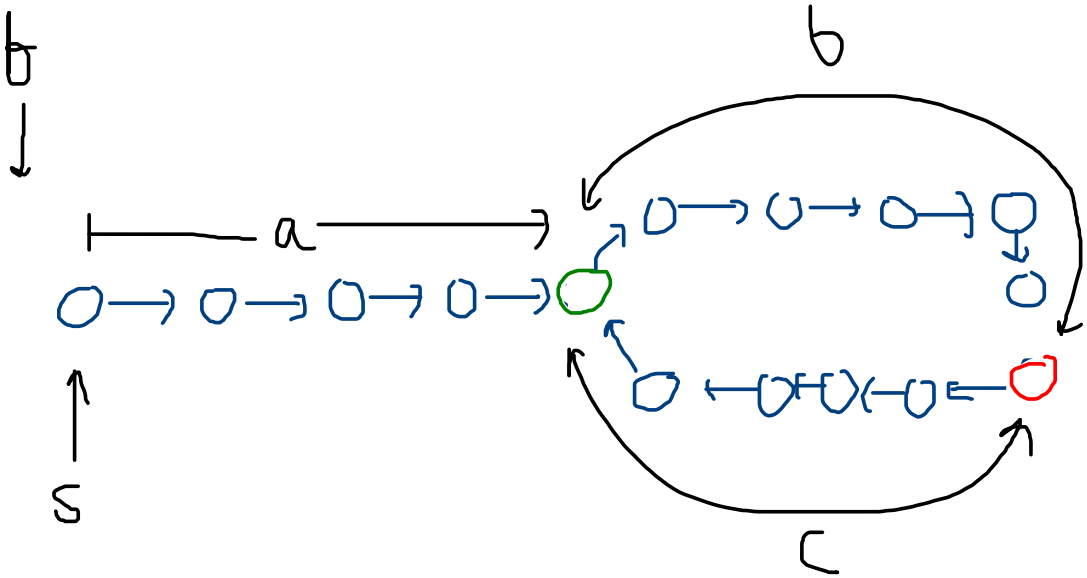
Cyclic point

fast and slow meets first time

a distance b/w head and green

b distance from green to red

c distance from red to green



d_s distance travel by S to reach red from head

d_f distance travel by F to reach red from head

n number of rotations S take before meeting of F pointer

m number of rotations F take before meeting of S pointer

ds = a + n x (b + c) + b

df = a + m x (b + c) + b

dist = velocity x time

time = dist/velocity

T = ds/s , T = df/f

$$\frac{d_s}{s} = \frac{d_f}{f}$$

$$df = \left(\frac{f}{s}\right) ds$$

let $\frac{f}{s} = r$

$$df = r\, ds$$

$$a + m(b+c) + b = r(a+b) + r \cdot n(b+c)$$

$$(b+c)\left[m - r \cdot n\right] = (a+b)(r-1)$$

$$\Rightarrow \boxed{a+b = \frac{(m - rn)(b+c)}{(r-1)}}$$

$$\Rightarrow \boxed{a+b = \frac{(m-n)(b+c)}{(r-1)}}$$

$$r-1 \neq 0$$

$$r-1 > 0$$

$$r > 1$$

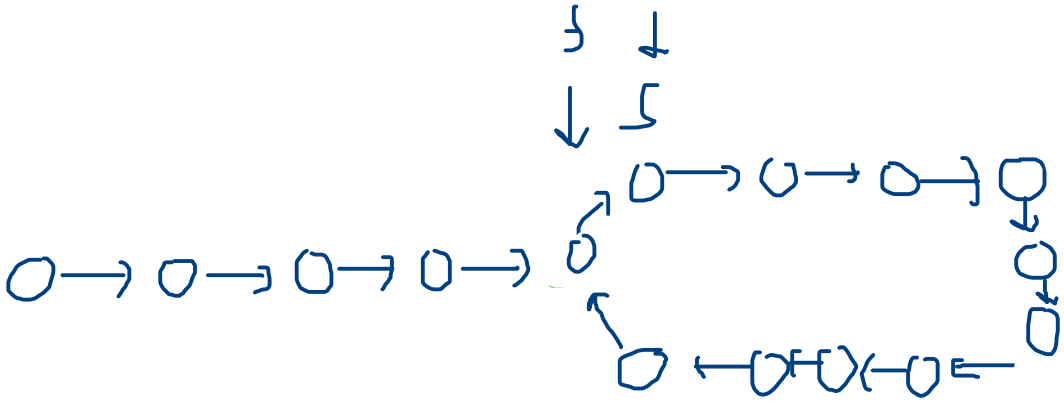
$$\frac{f}{s} > 1$$

$$\boxed{f = ps}$$

$$\boxed{p \in \mathbb{R}^+, p \neq 1}$$

$m = 1$

$n = 1$



f = p x s, p = 2, p -> R+, p != 0

$f = PS$

$f = 3$

$S = 2$

best --> s will cover total of 0 rotation

$S = 2S$

$$a + b = \frac{(m - 2n)(b + c)}{(n - 1)}$$

r = 2

$$a + b = (m - 2n)(b + c)$$

$$a + b = (b + c)$$

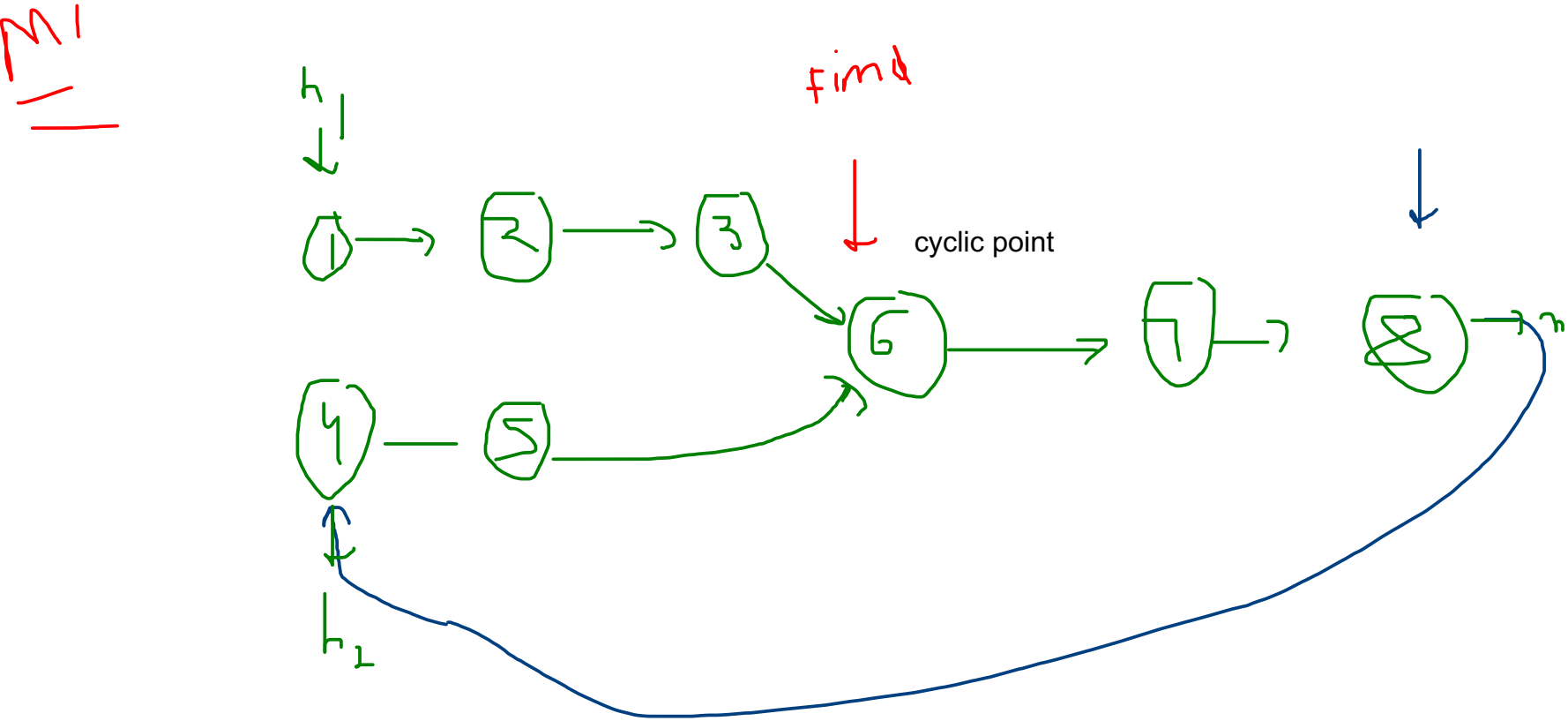
m - 2n

$$a + \cancel{b} = \cancel{b} + c$$

a = c

dist from head to green and red to green is equal

Intersection of LL



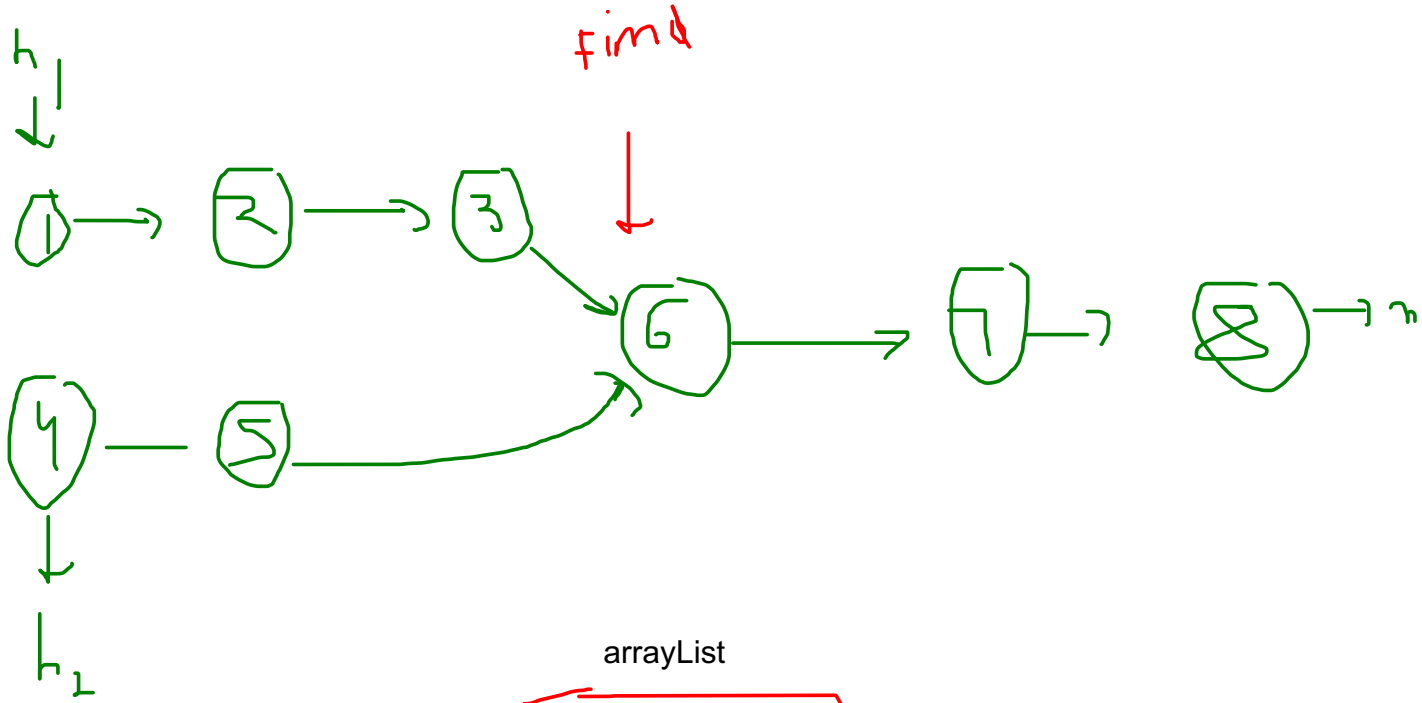
input

head1 and head2

correct input again

Intersection of LL

M2



input

head1 and head2

arrayList

arr1 = 1, 2, 3, 6, 7, 8

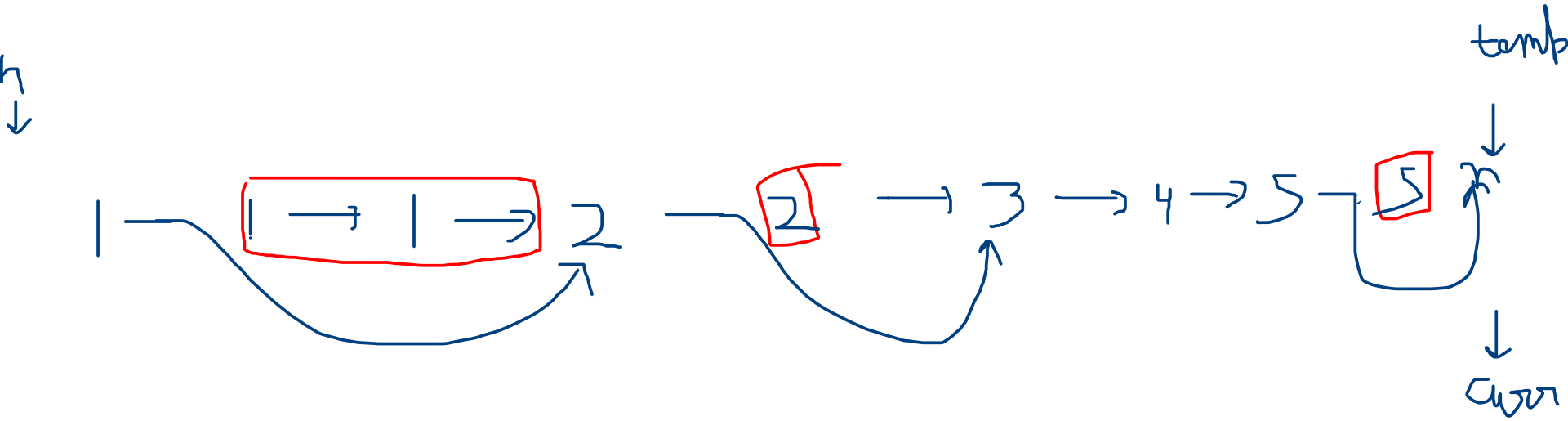
arr2 = 4, 5, 6, 7, 8

Remove Duplicates

h

$1 \rightarrow 1 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 5$

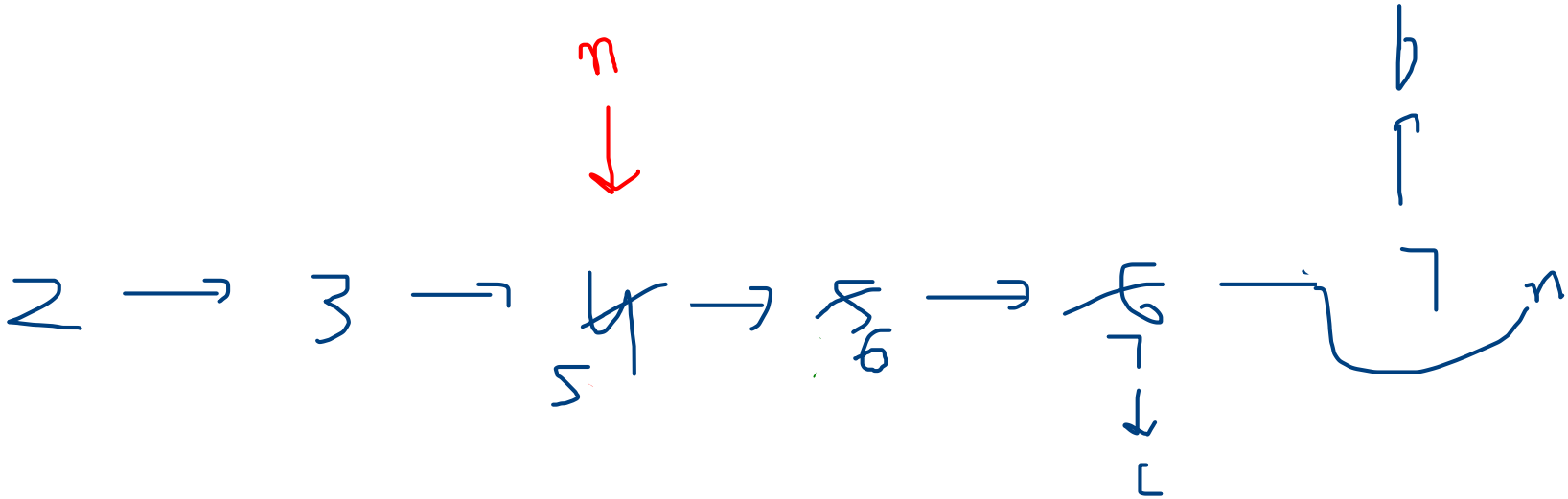
$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$



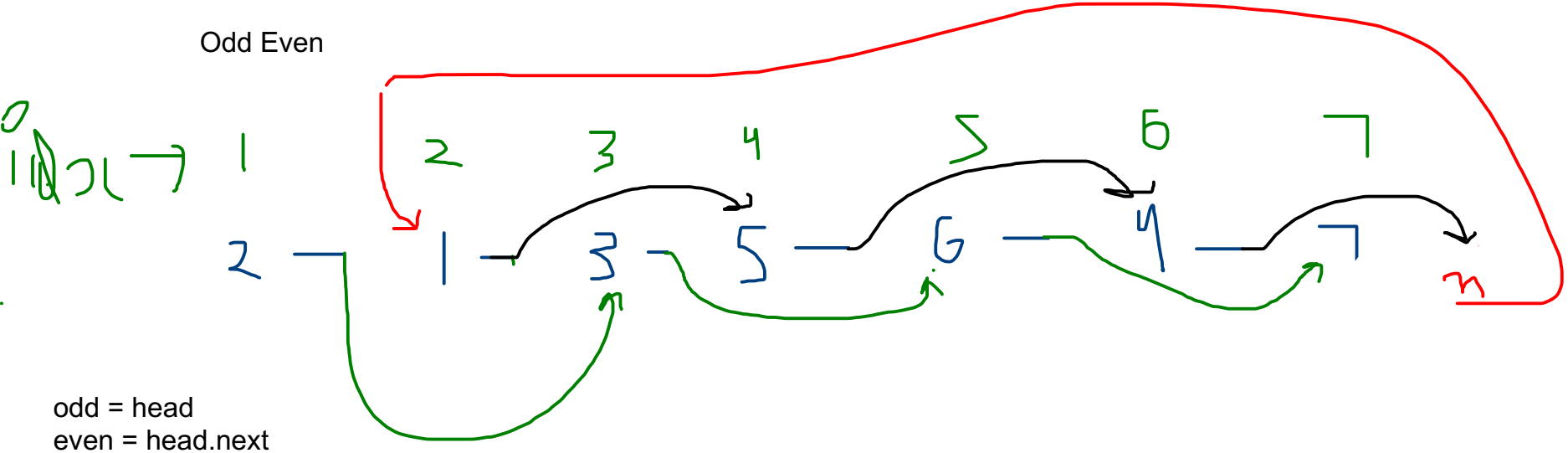

```
curr = head

while(curr != null) {
        temp = curr
}
```

remove a node in LL

head is not given,
remove node is given



Odd Even



odd = head
even = head.next

odd.next = even.next

odd = odd.next

even.next = odd.next

even = even.next

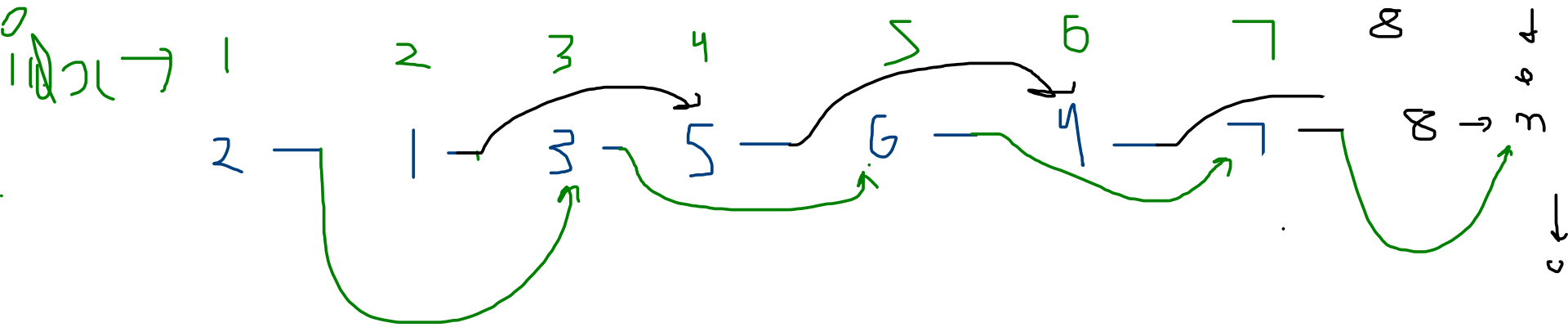
[2,1,3,5,6,4,7]

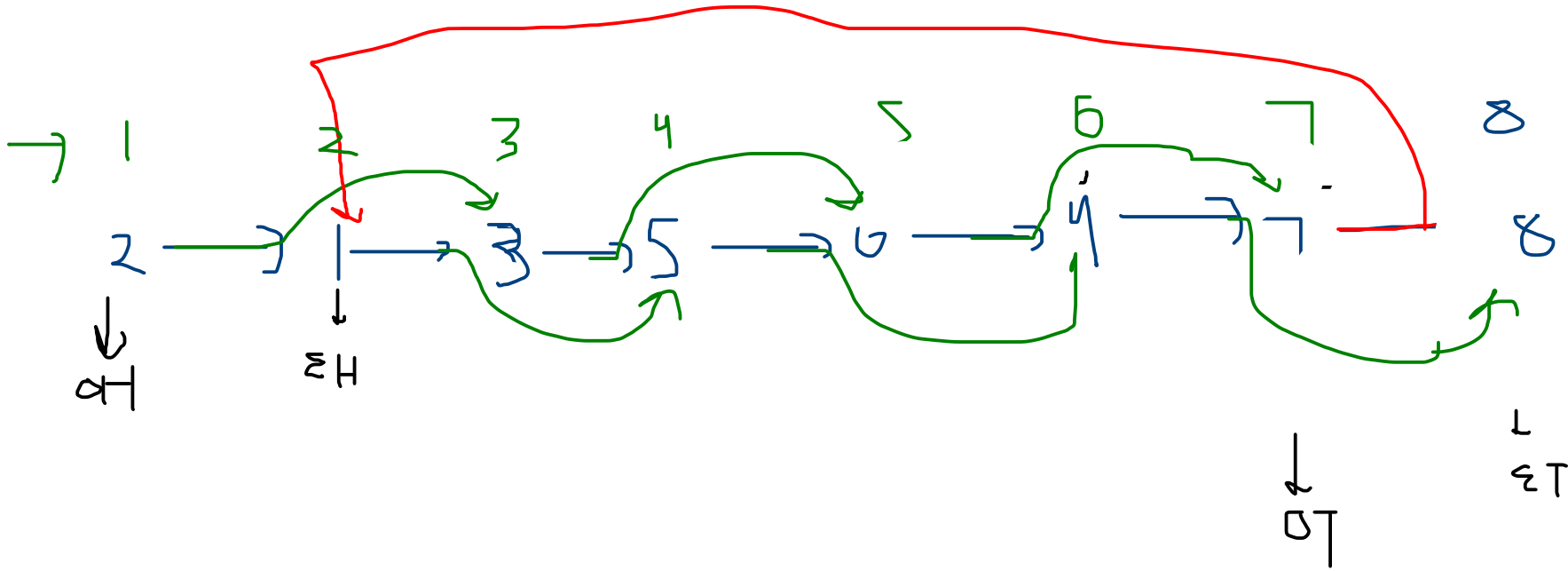
[2,3,6,7,1,5,4]

odd

even

2 → 3 → 6 → 7 → 1 → 5 → 4


0
1
2
3
4
5
6
7
8
2
1
3
5
6
4
7
8
n



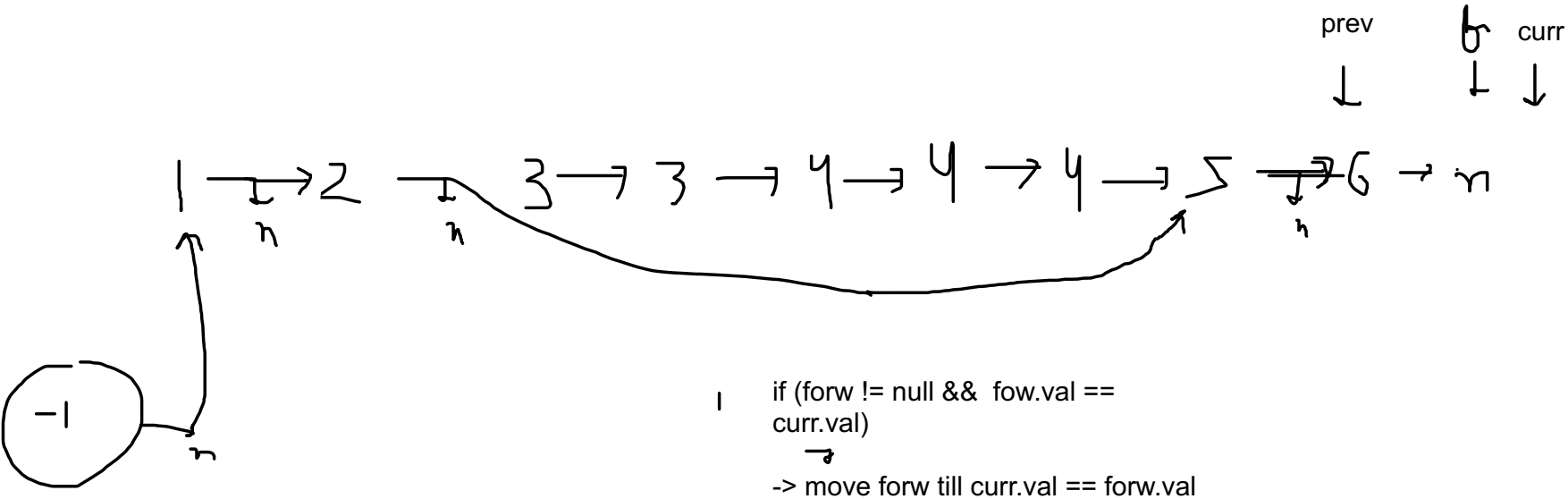

```
oddT = head
evenT = head.next

while(even != null && even.next != null)
oddT.next = evenT.next

oddT = oddT.next

evenT.next = oddT.next

evenT = evenT.next
```

Remove Duplicates II LinkedList

till prev LL do not contain any Duplicates

dummy = new node
prev = dummy
curr = head

Case1



dummy
node

1 if (forw != null && fow.val == curr.val)

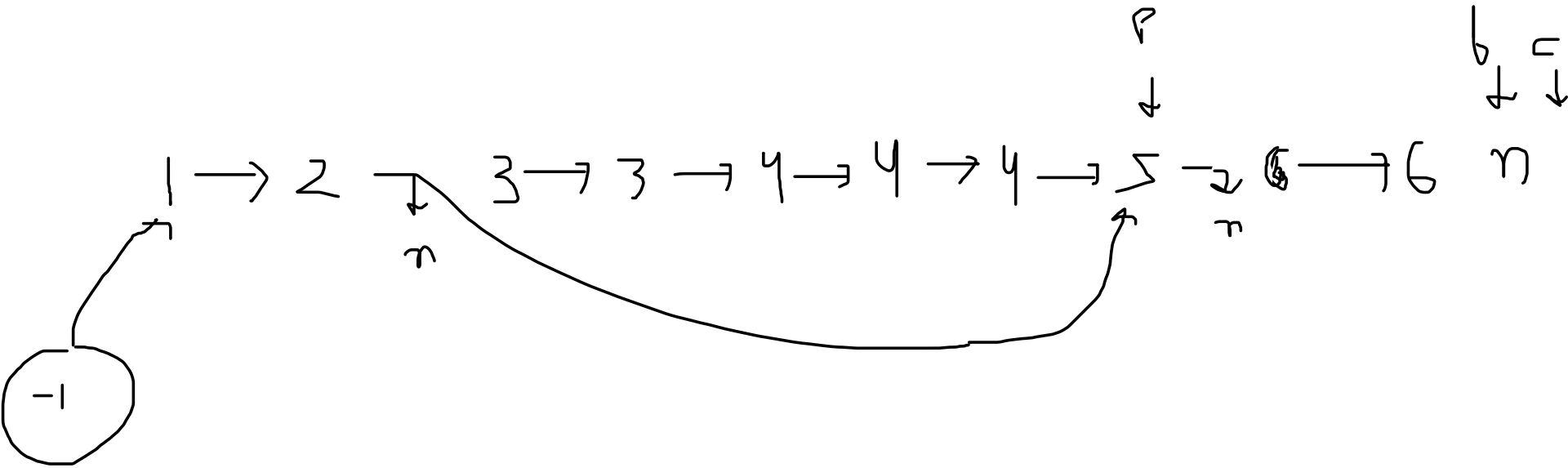
-> move forw till curr.val == forw.val
--> curr.val != forw.val --> curr = forw

2 if (fow.val != curr.val)

-> prev.next = curr
-> curr.next = null
-> prev = curr
-> curr = forw

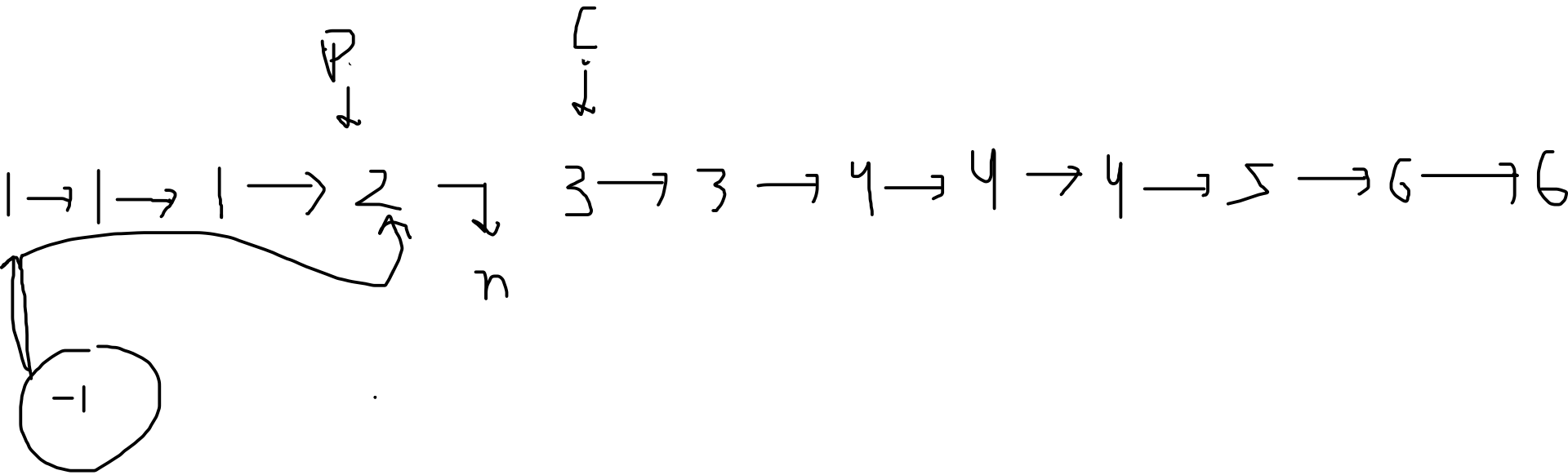
return dummy.next

Case2



dummy
node

Case3



dummy
node

Reverse in K Groups

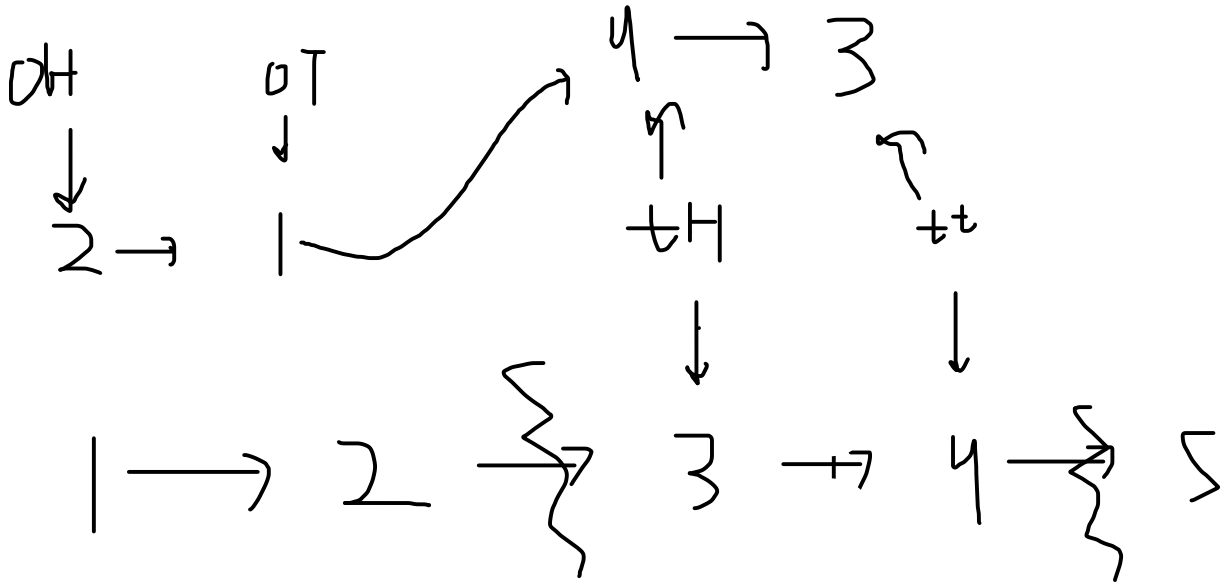
1 → 2 → 3 → 4 → 5 → 6

k
3

3 → 2 → 1 → 6 → 5 → 4

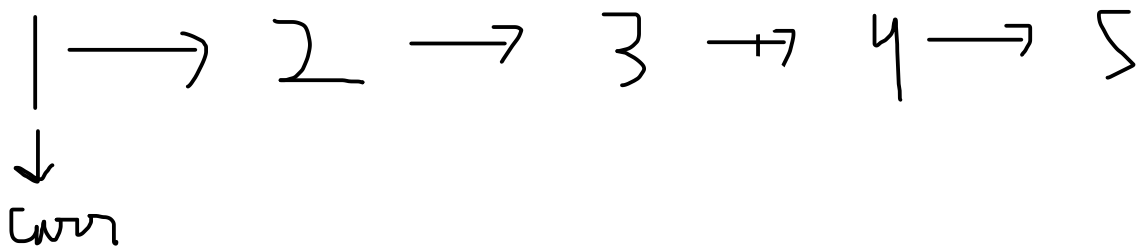
k
4

4 → 3 → 2 → 1 → 5 → 6



k = 2

k=2



count total Nodes = 6

curr = head

tempCount = k = 2

while tempCount > 0

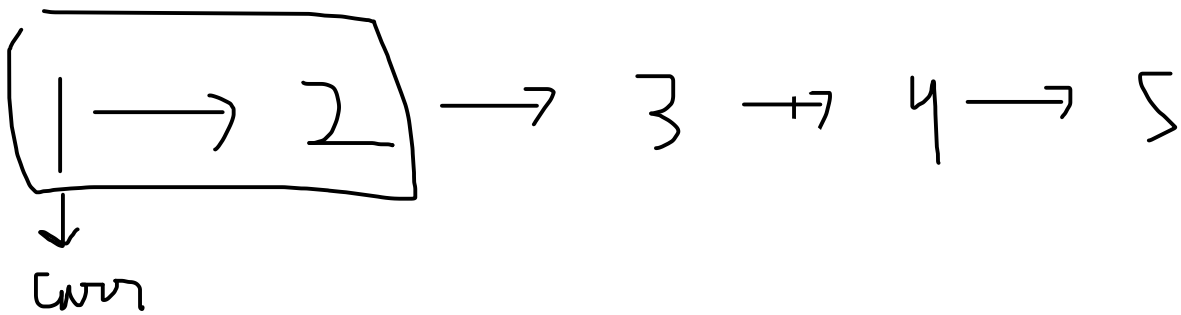
orig --> my ans --> reversed in k groups

origHead = null
origTail = null

temp --> to reverse part of list which k nodes

tempHead = null
tempTail = null

k=2



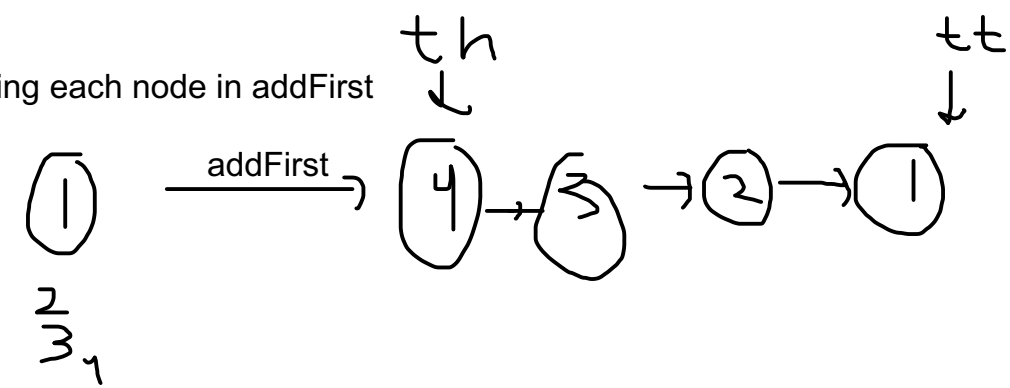
4

count total Nodes = 6

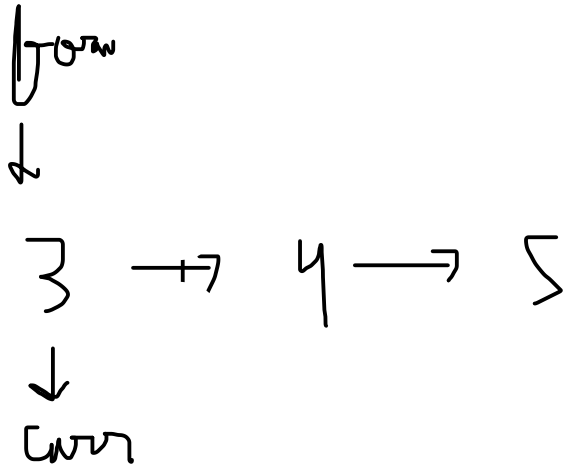
curr = head

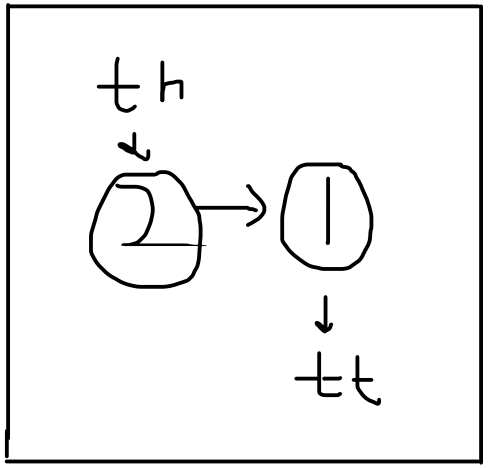
tempCount = k = 2

reverse them by passing each node in addFirst



k=2





Reversed k nodes

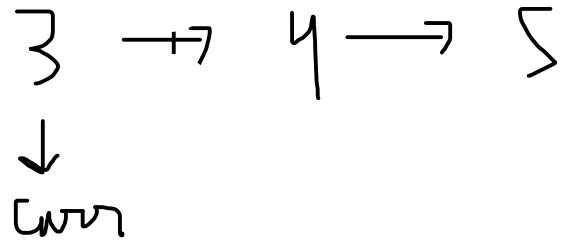
count total Nodes = 5

curr = head

while(curr != null)

--> tempCount = k = 2 4 0
tempCount > 0 -->
curr.next = null
addFirst(curr)
curr = forw

k=2



count total Nodes = 5

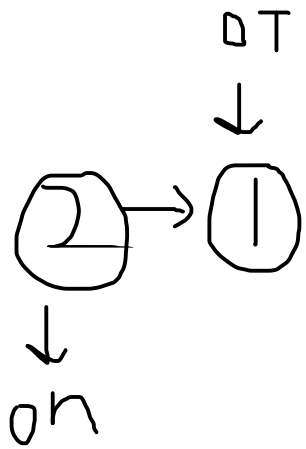
curr = head

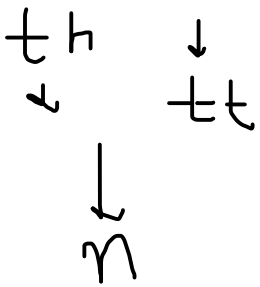
while(curr != null)

--> tempCount = k =2 4 0
tempCount > 0 -->
curr.next = null
addFirst(curr)
curr = forw

if (oh == null) --> oh = th, ot = tt
if (oh != null) -->

th = null
tt = null





k=2

count total Nodes = ~~5~~ 3

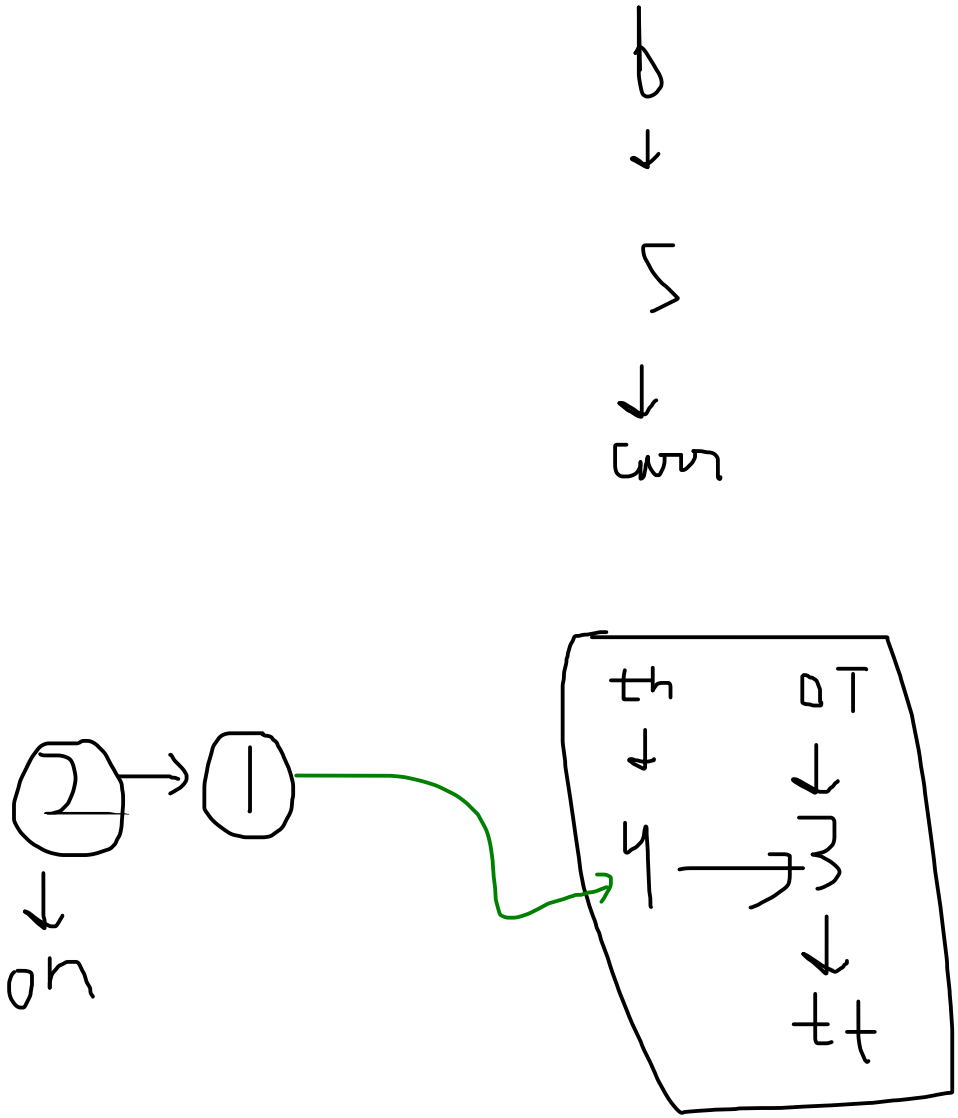
curr = head

while(curr != null)

--> tempCount = k = ~~2~~ + 0
tempCount > 0 -->
curr.next = null
addFirst(curr)
curr = forw

if (oh == null) --> oh = th, ot = tt
if (oh != null) --> ot.next = th, ot = tt

th = null
tt = null



k=2

count < k

return oh

count total Nodes = ~~5~~ 3 1

curr = head

while(curr != null)

--> tempCount = k = 2
tempCount > 0 -->
curr.next = null
addFirst(curr)
curr = forw

if (oh == null) --> oh = th, ot = tt
if (oh != null) --> ot.next = th, ot = tt

th = null
tt = null