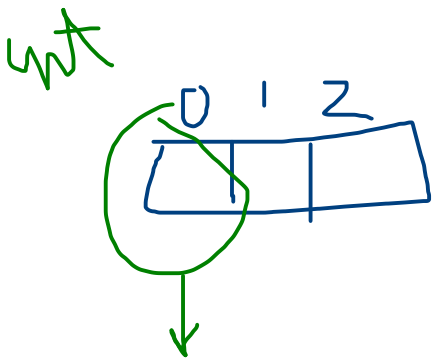
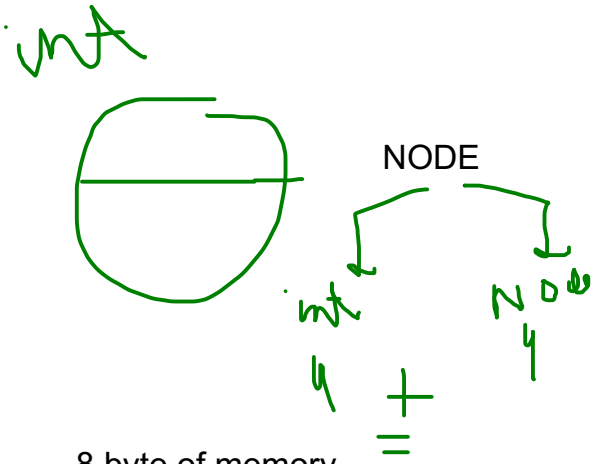
array



4 byte of memory

linkedList

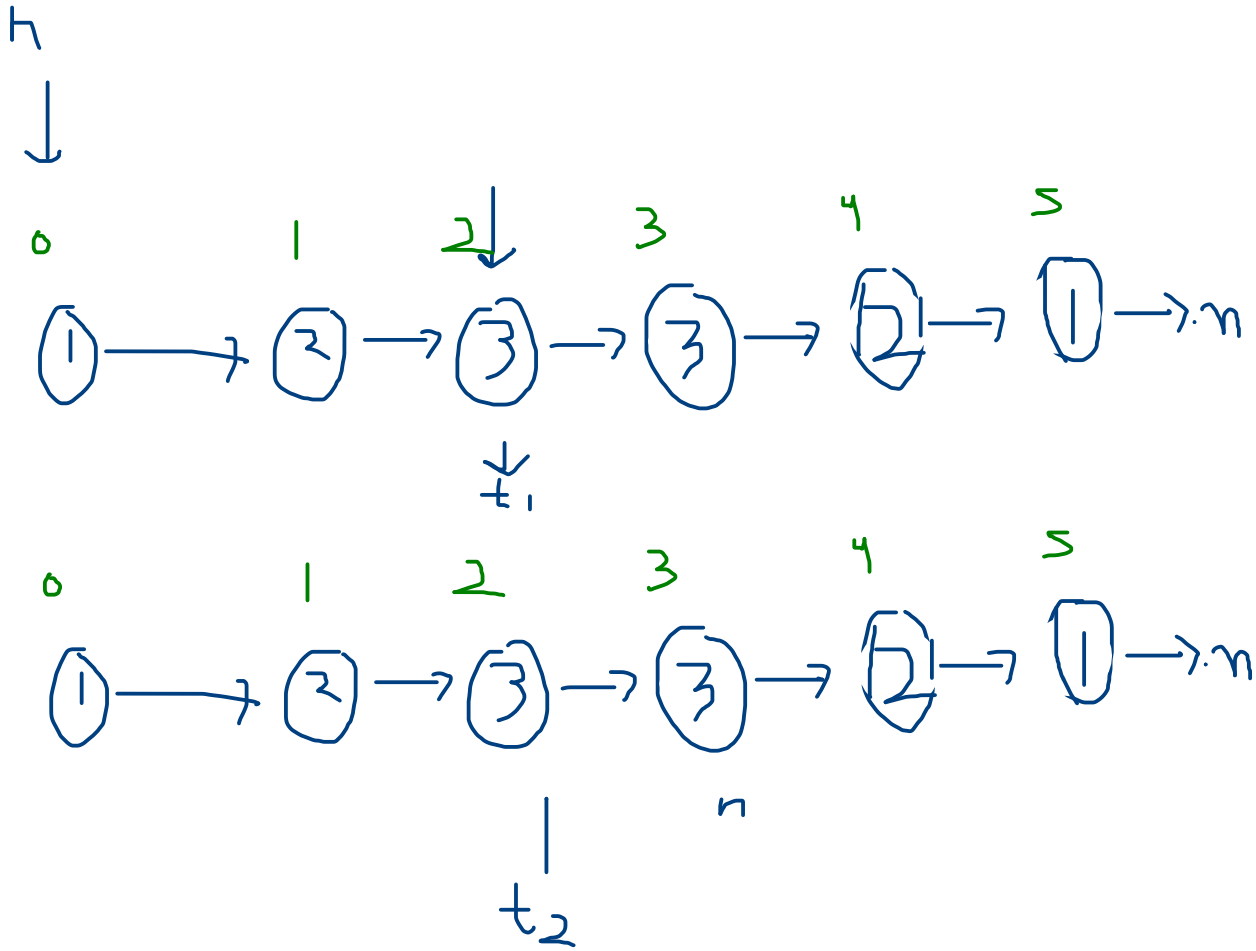


8 byte of memory

single cell of arr memory < singly node of LL

isPalindrome LL

Method1

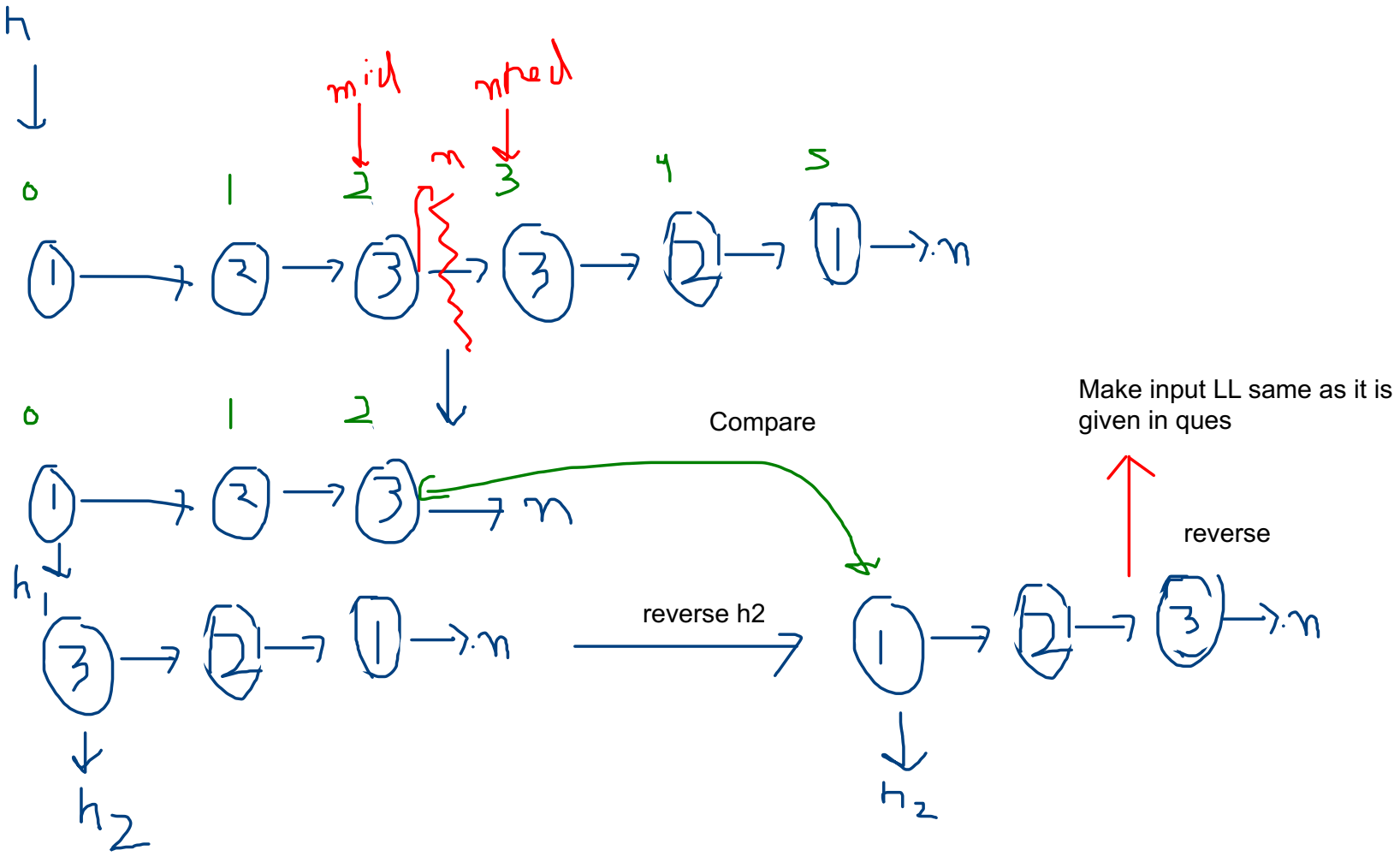


compare till < n / 2

new LL
Reverse

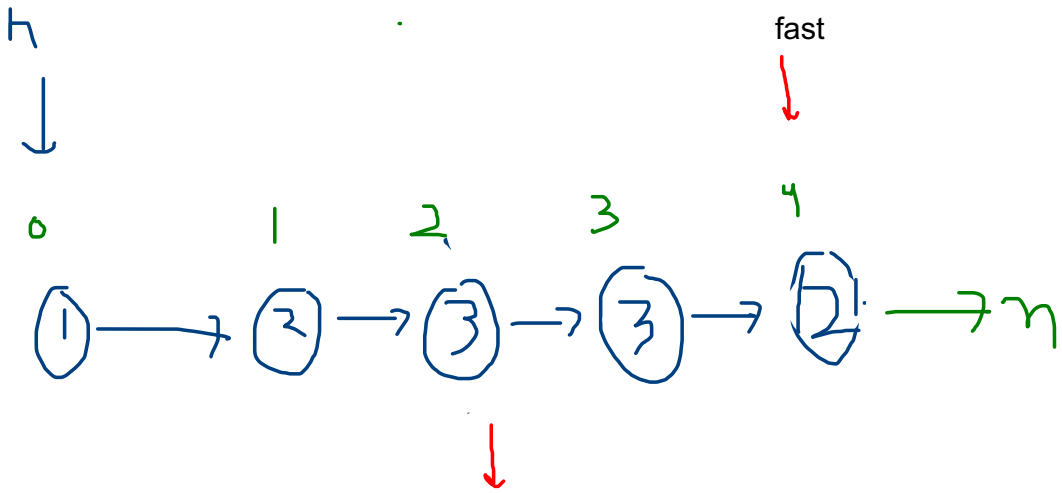
Method 2

middle of LL



Middle of LL

Case1: odd length --> one mid



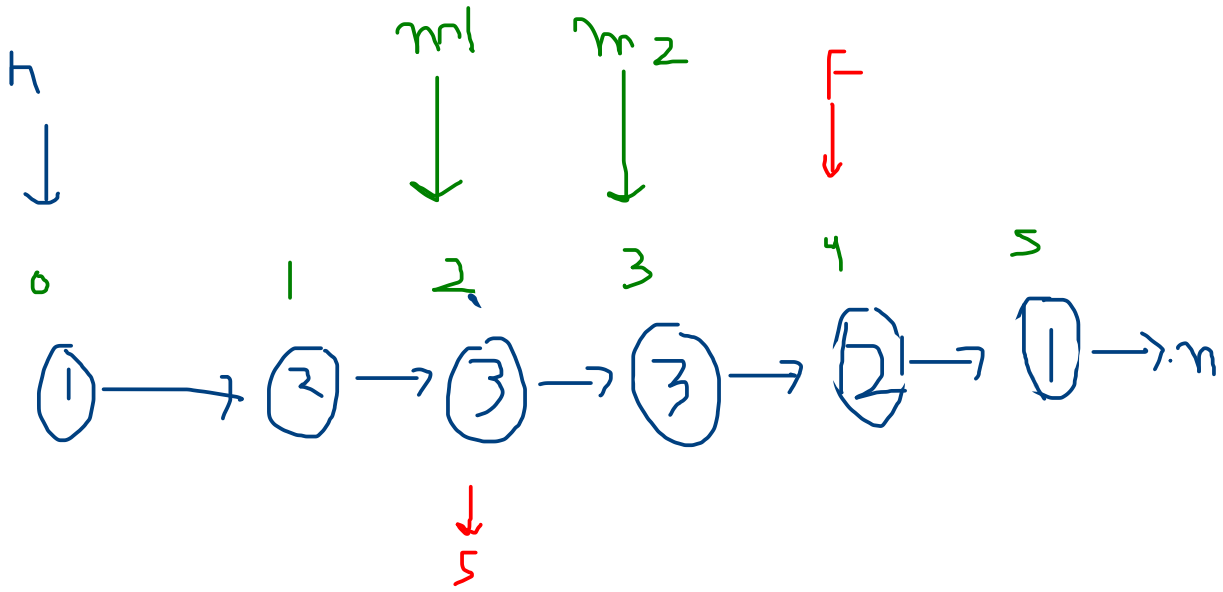
move

s = s.next
f = f.next.next

f.next == null --> stop

slow is middle

Case2: even length --> 2 mid



solving ques --> use M1

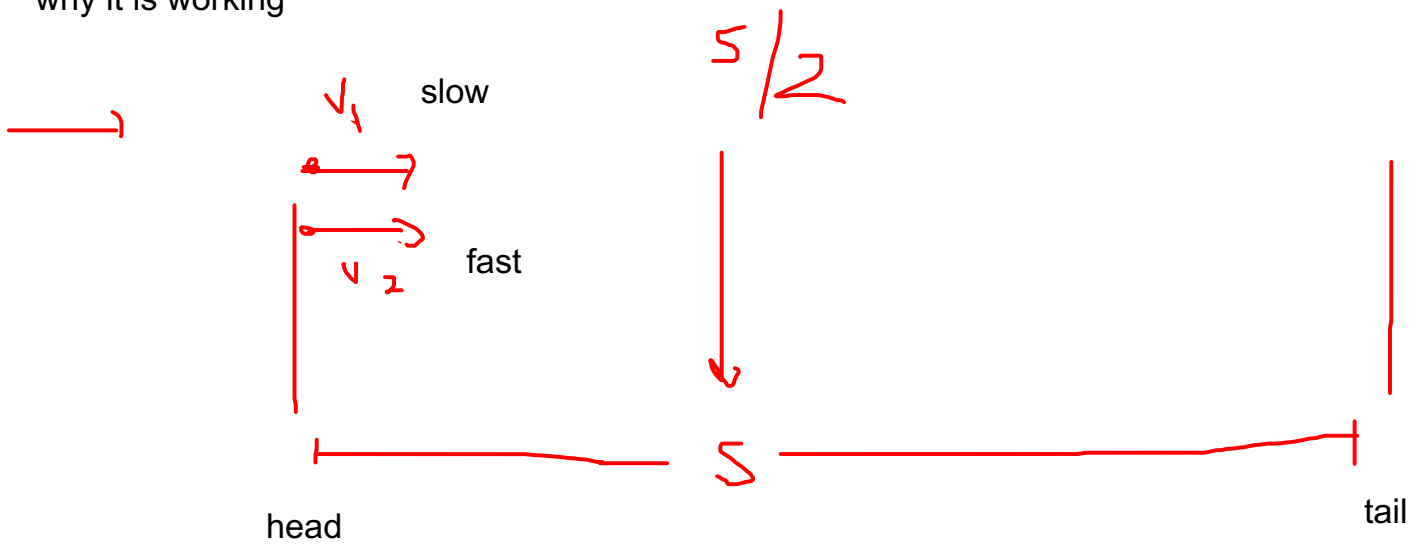
only need to find mid --> ans depends on ques but by default give m2

f.next.next = null -- for M1

f == null --> for M2

1

why it is working



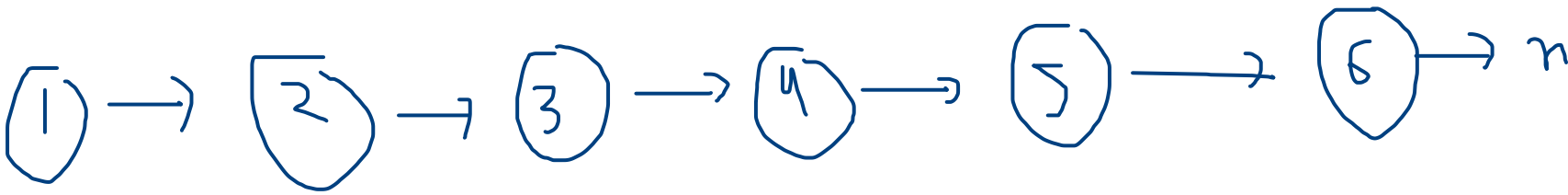
dist = velo x time

$$\frac{S}{V_2} = \frac{\frac{S}{2}}{V_1} = \frac{S}{2V_1}$$

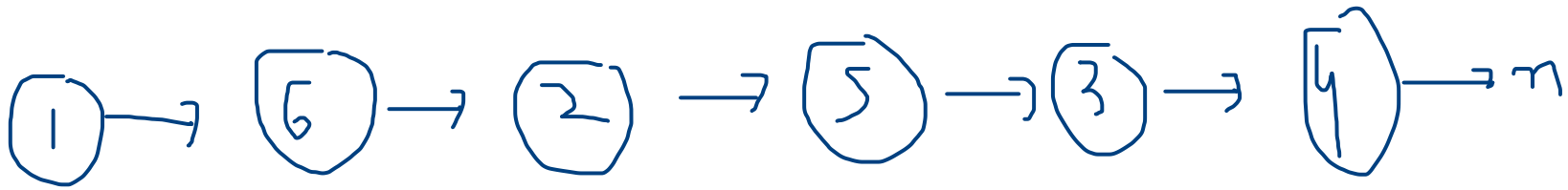
$$V_1 = \frac{V_2}{2}$$

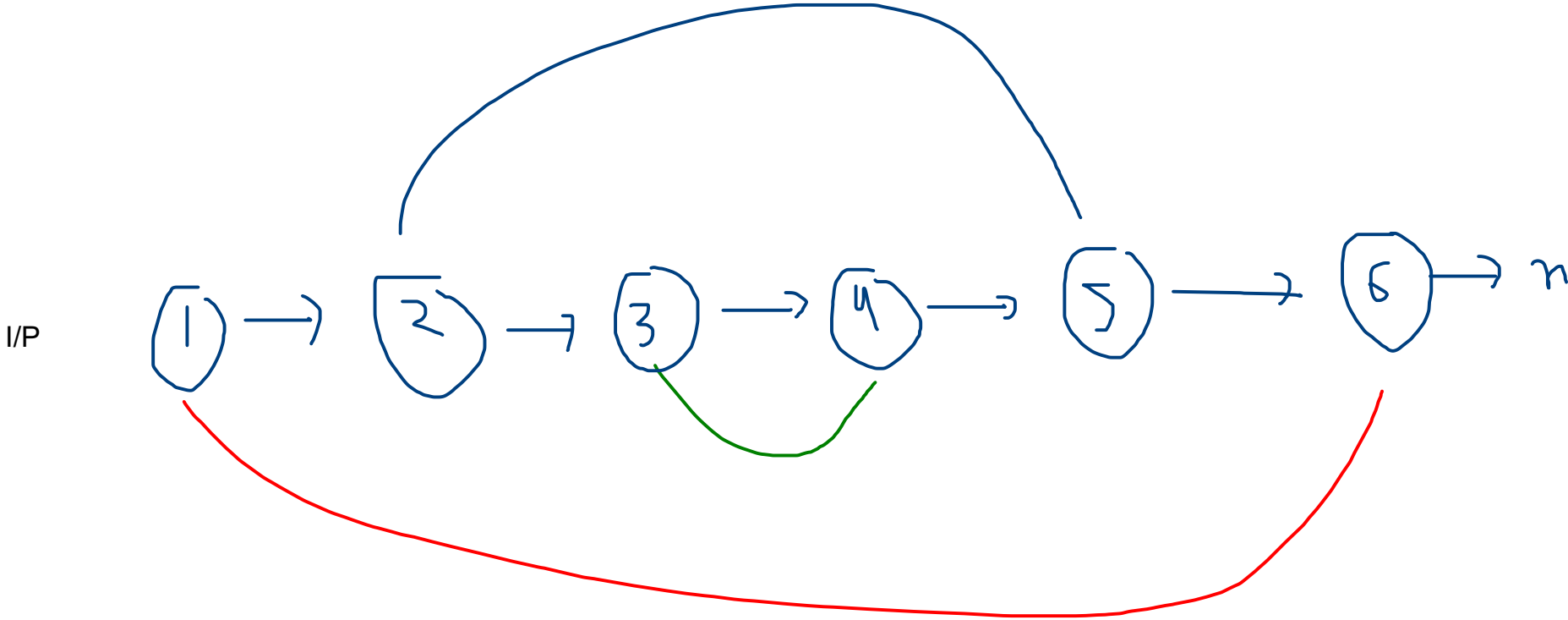
FOLD of LL

I/P

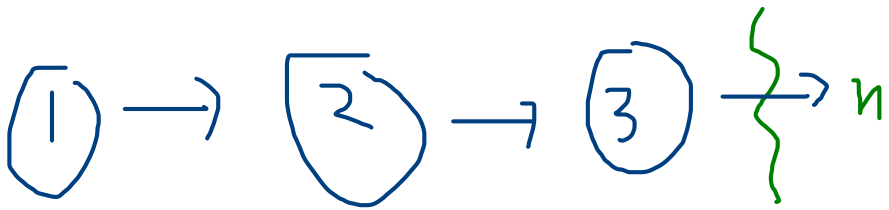


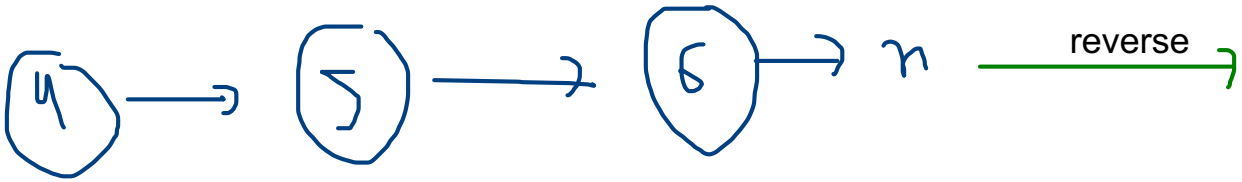
O/P

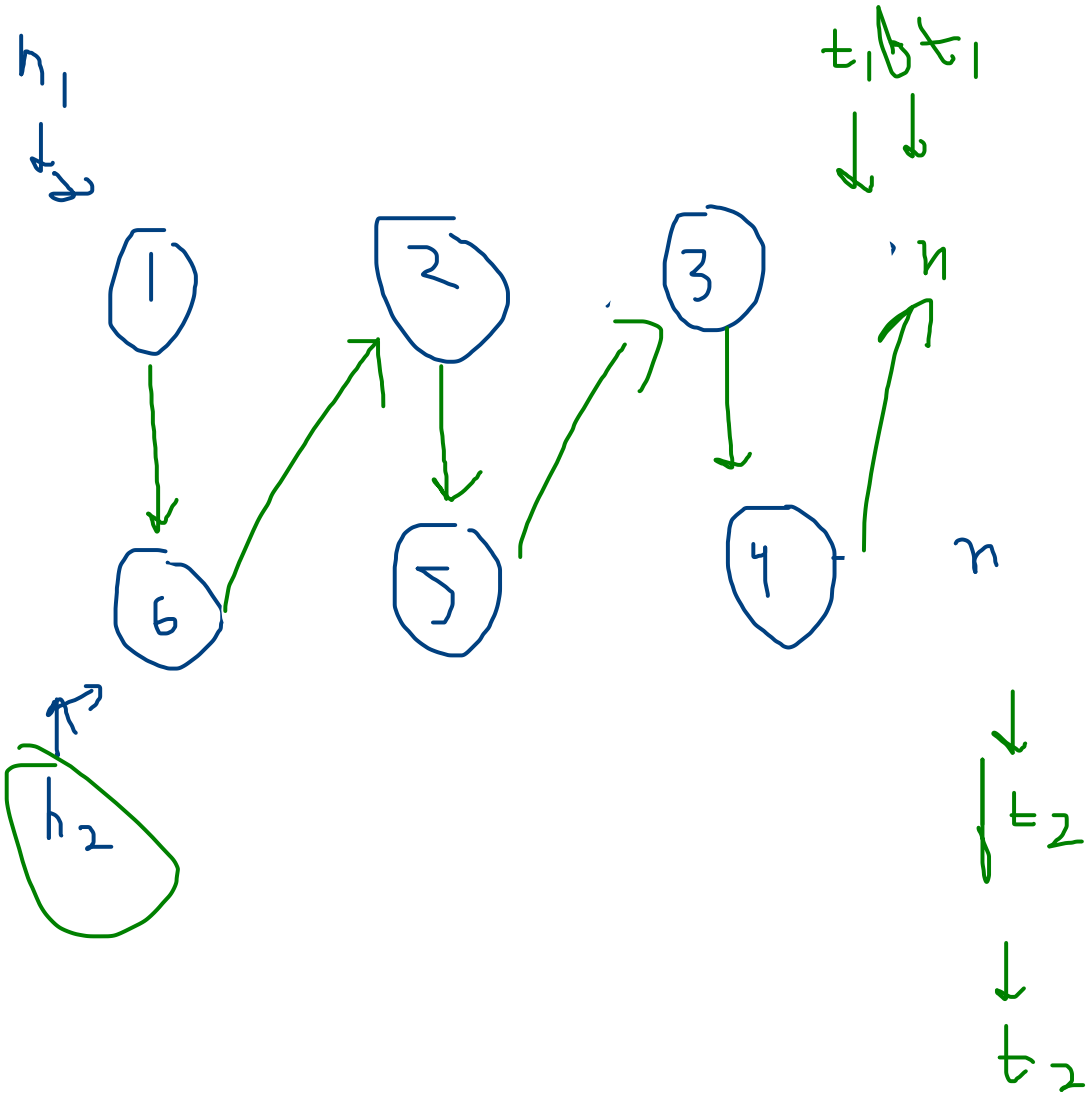



I/P
1
2
3
4
5
6
n

I/P





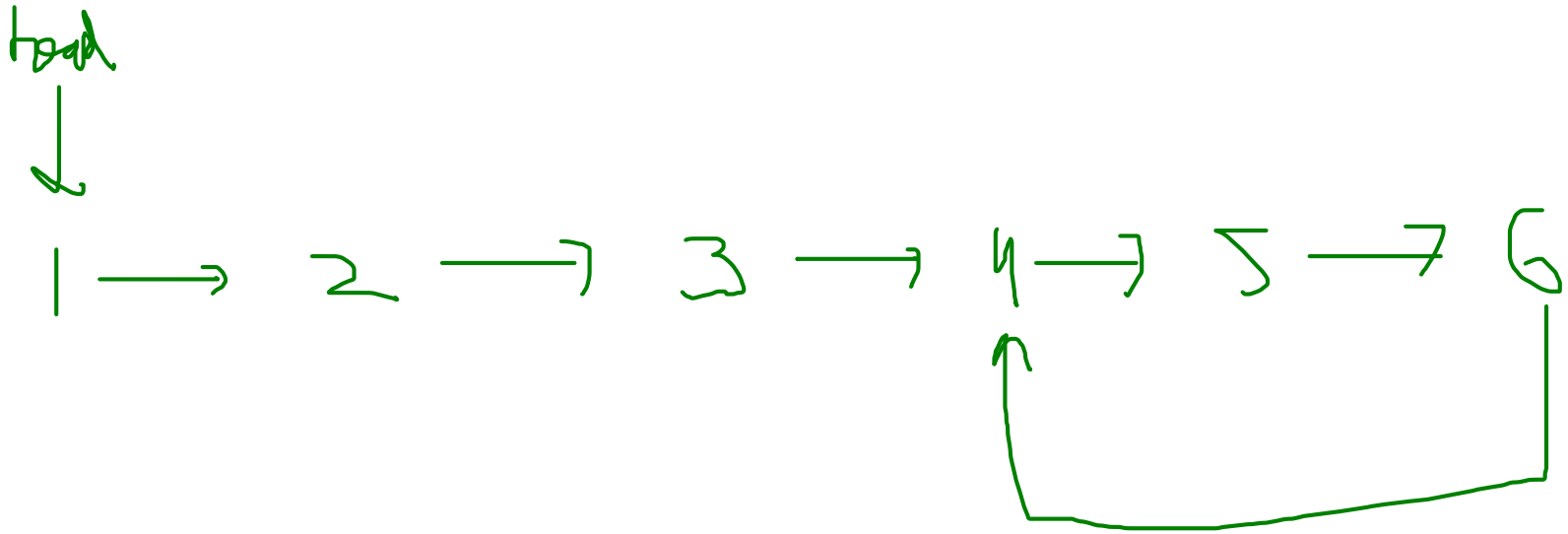


t1 != null && t2 != null

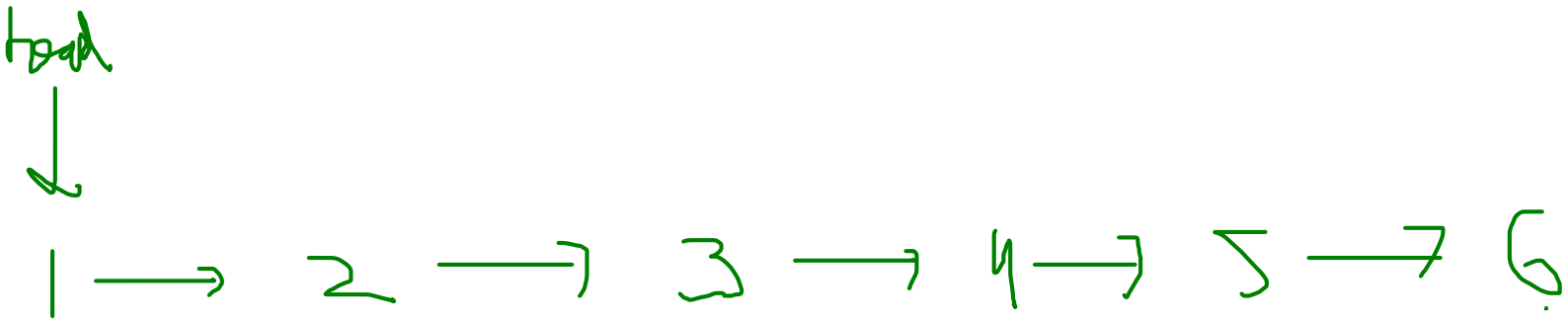
ft1 = t1.next
ft2 =t2.next;
t1.next = t2
t2.next = ft1

t1= ft1
t2 = ft2

Cycle in LinkedList

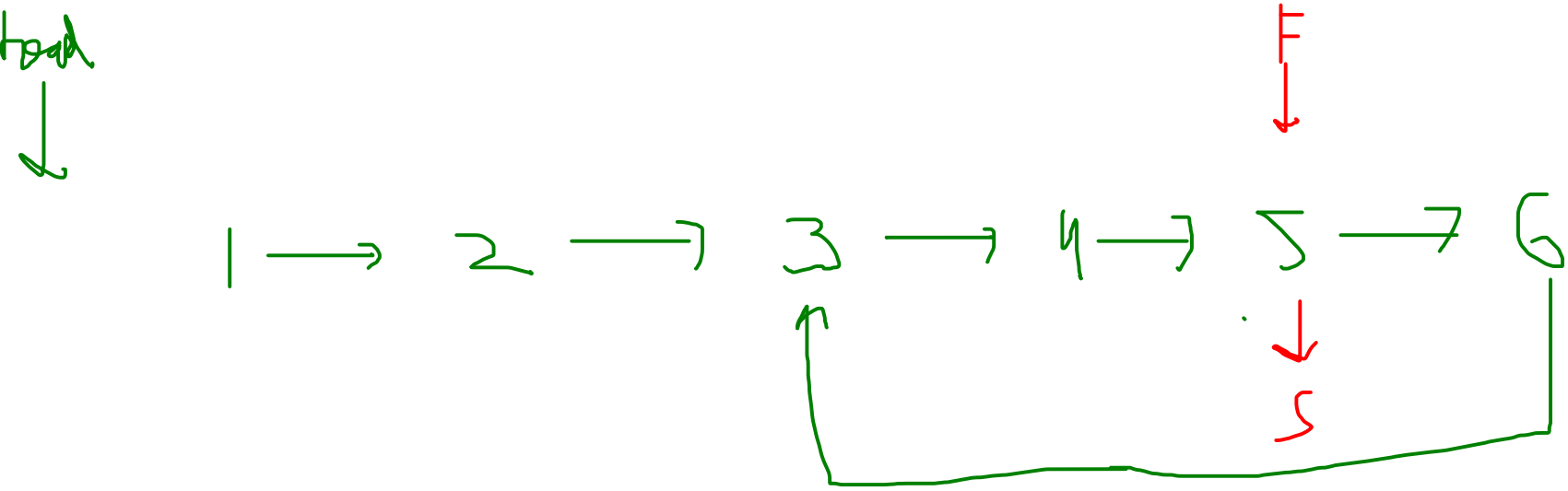


o/p = true



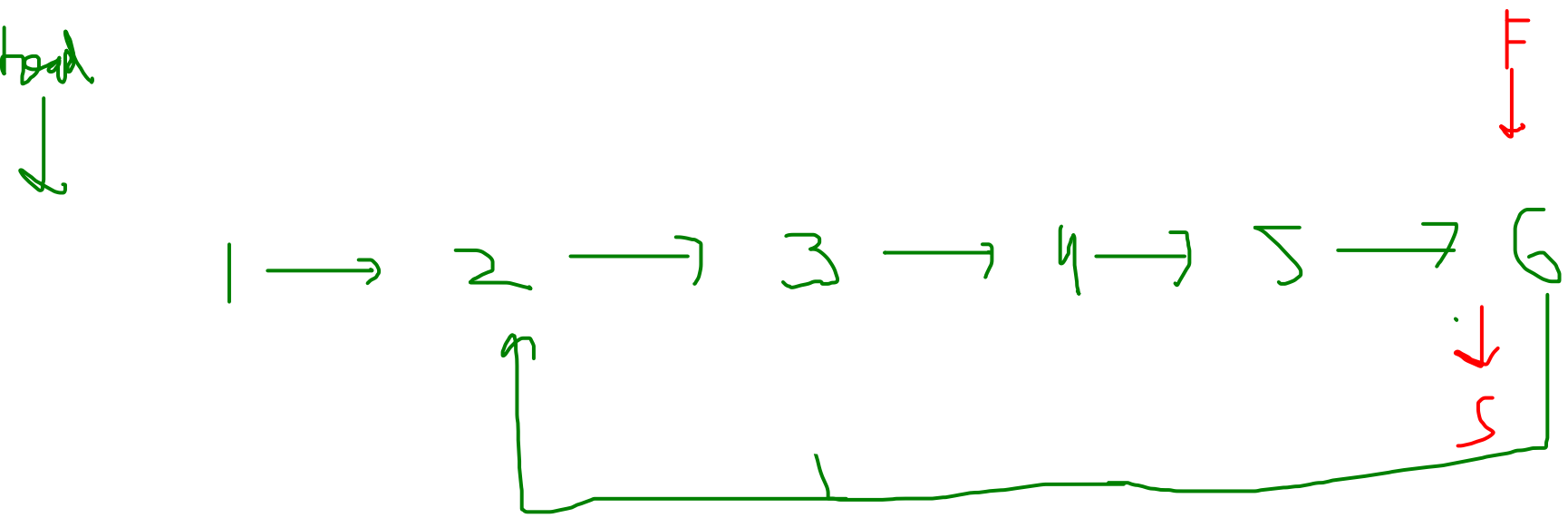
o/p false

Floyd's Cycle Detection Algo

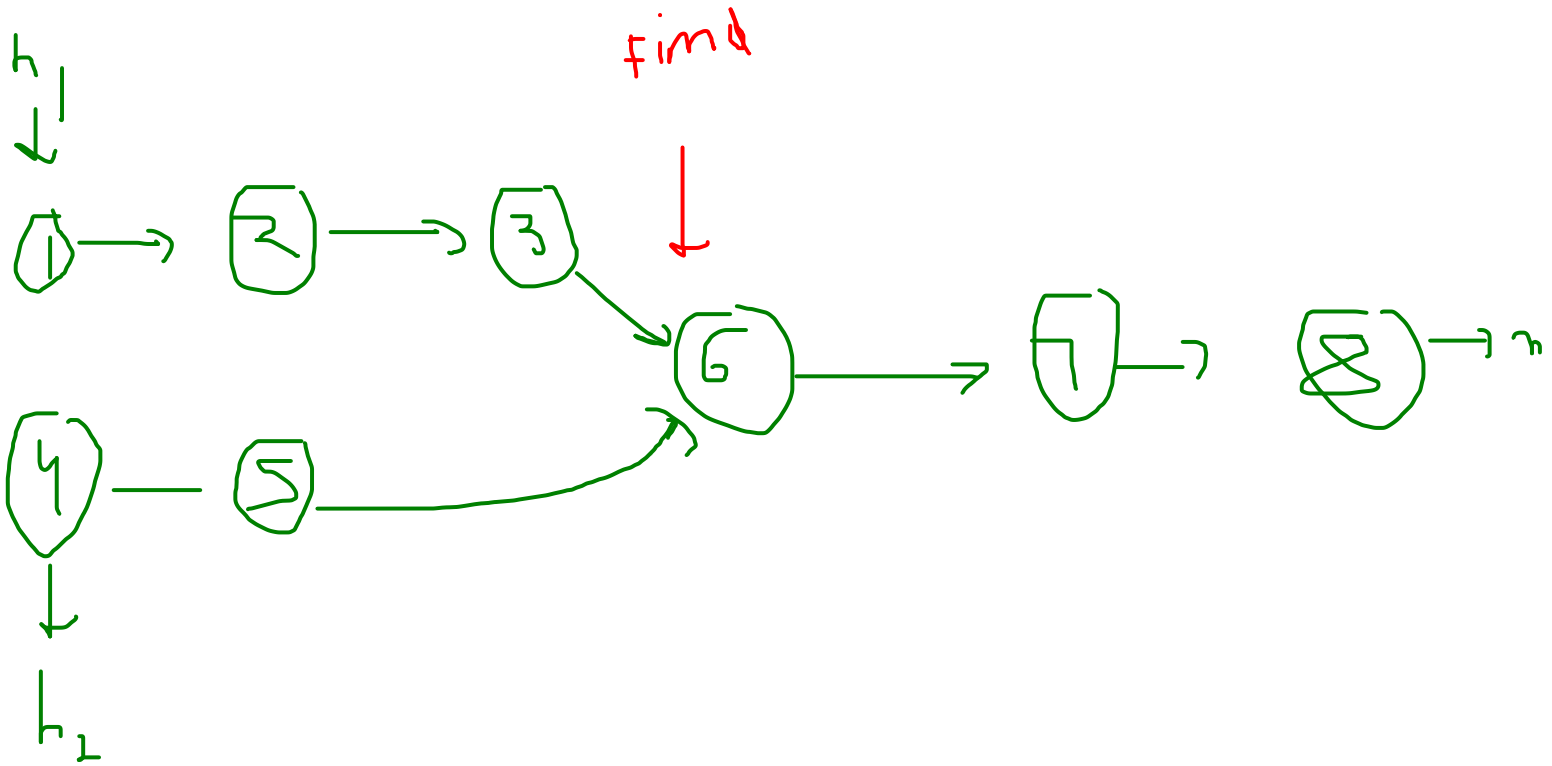


s = s.next
f = f.next.next

f == s --> list have cycle


head
F
1
2
3
4
5
6
5

Intersection of LL



input

head1 and head2