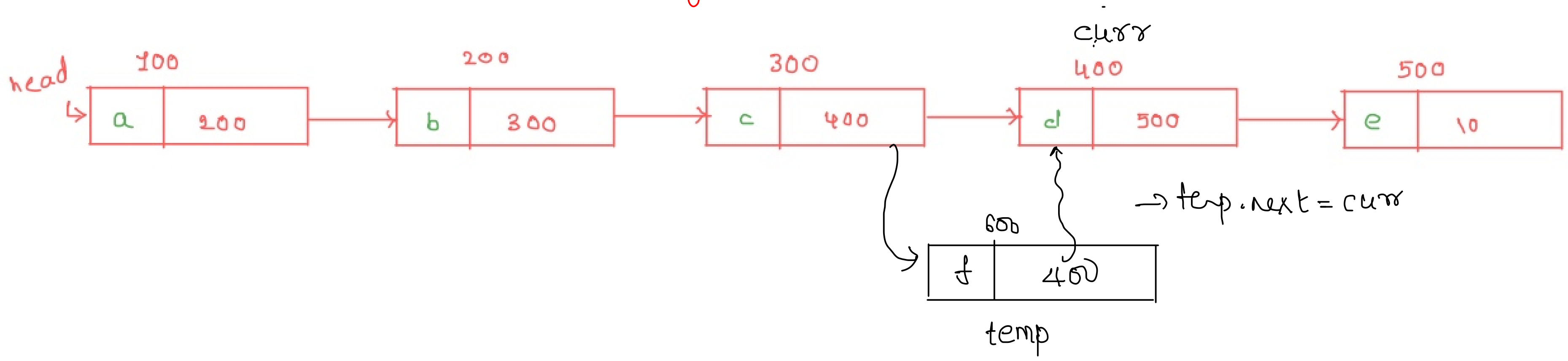
a → b → c → f → d → e

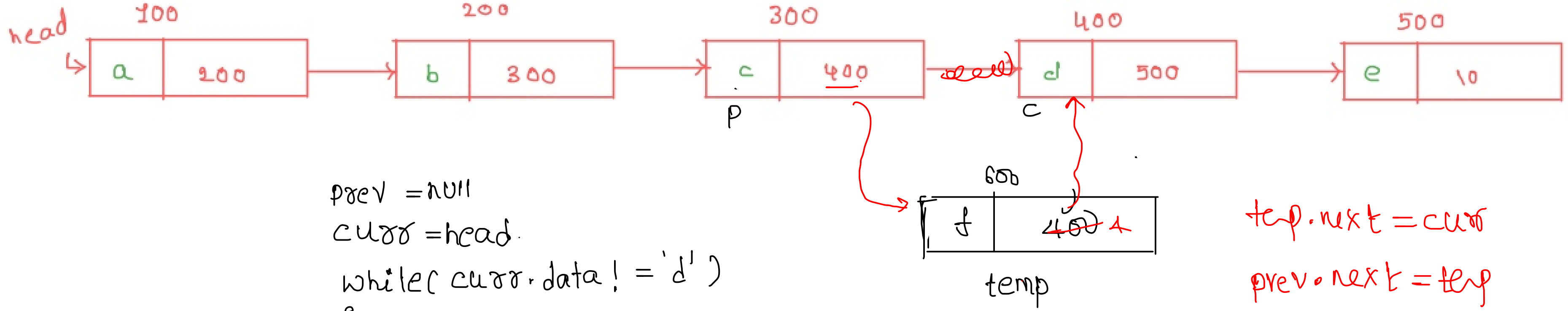
4) Adding an Element before a particular element

(f)

d
↳ Always present.



→ temp.next = curr

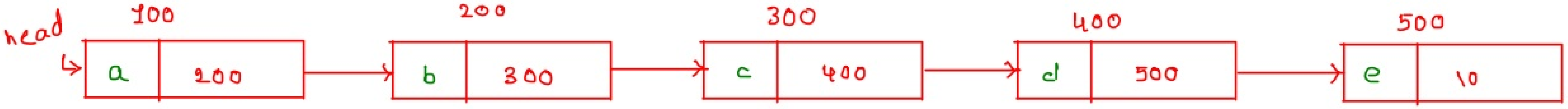


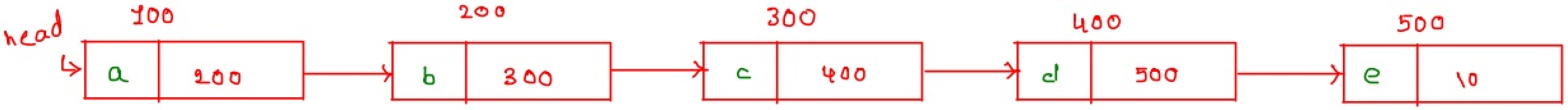
```
prev = null
curr = head
while( curr.data != 'd' )
{
        1. prev = curr;
        2. curr = curr.next;
}
```

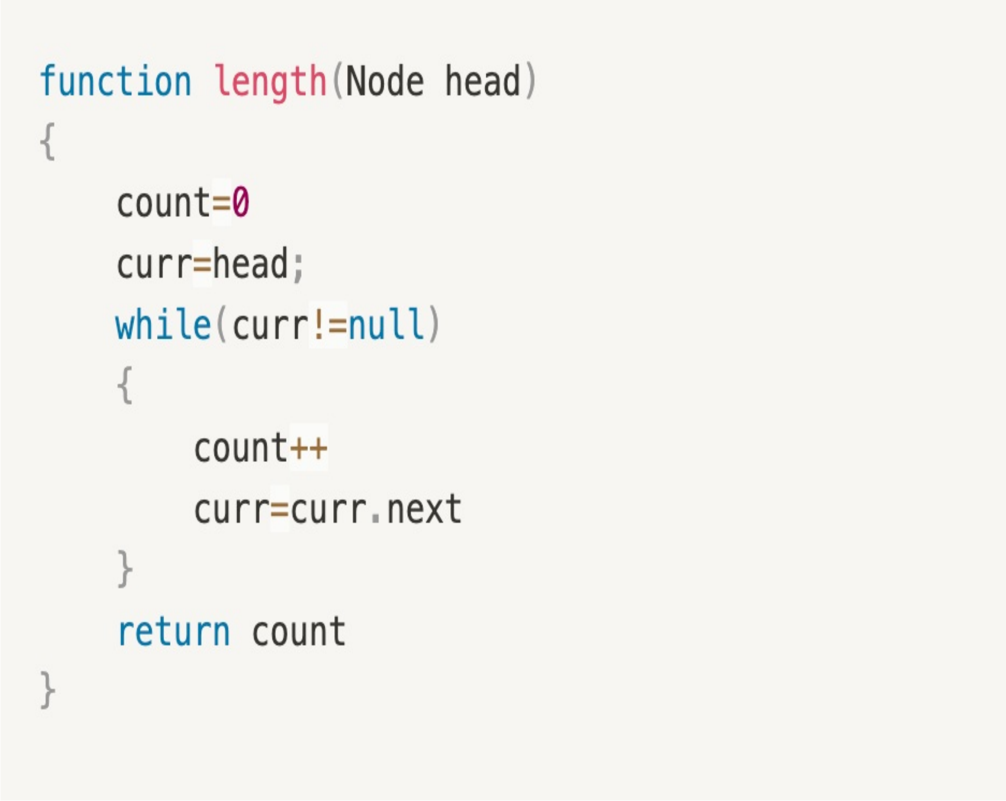
temp.next = curr

prev.next = temp

1) Find the length of SLL

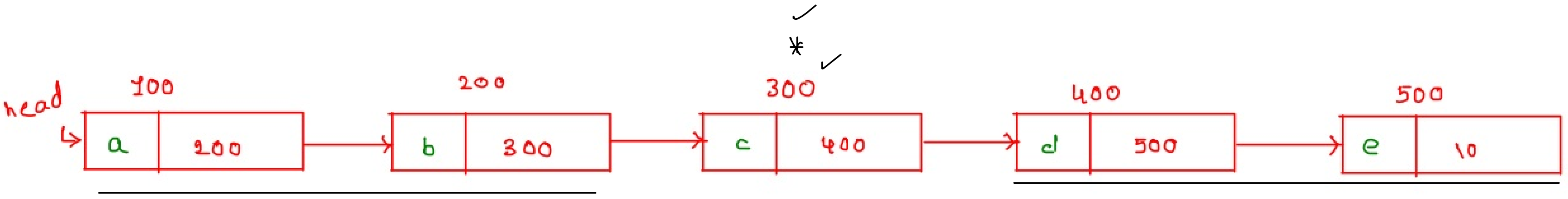






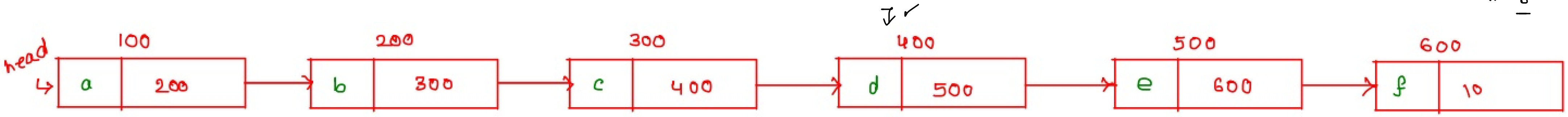
```
function length(Node head)
{
    count=0
    curr=head;
    while(curr!=null)
    {
        count++
        curr=curr.next
    }
    return count
}
```

2) Find the Middle node in a SLL [Adobe, Amazon, Flipkart, GE, Microsoft, Qualcomm, Samsung, VMWare, Wipro, Zoho]



$n = 5$ odd.

$\frac{n}{2} + 1 = \textcircled{3}$



$n = 6$

$\frac{n}{2} + 1 = \textcircled{4}$

Ap1 :-

1) find length (n)

2) n/2 ✓

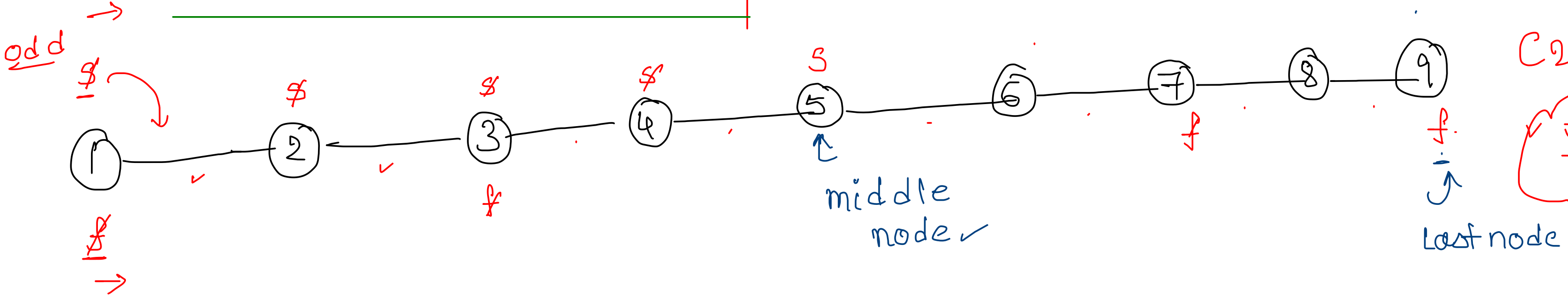
Ap2

without finding length

2ptr ✓

f = f.next.next ✓

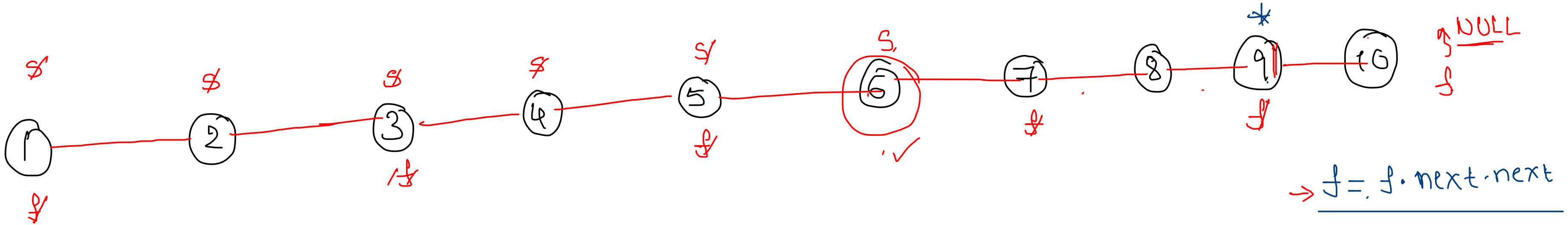
→ NPE



C2

f.next != NULL

F



→ f = f.next.next

null

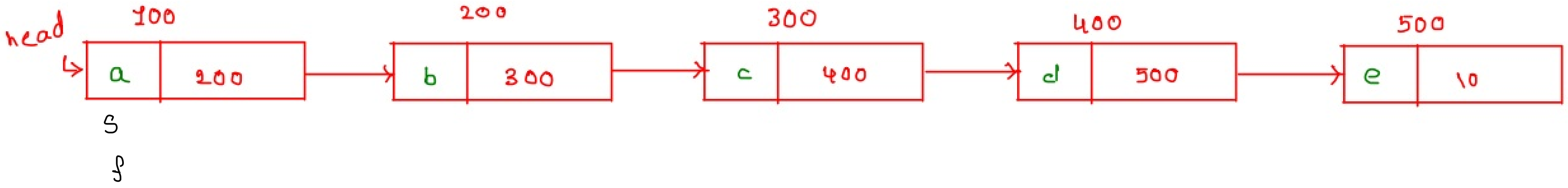
NPE

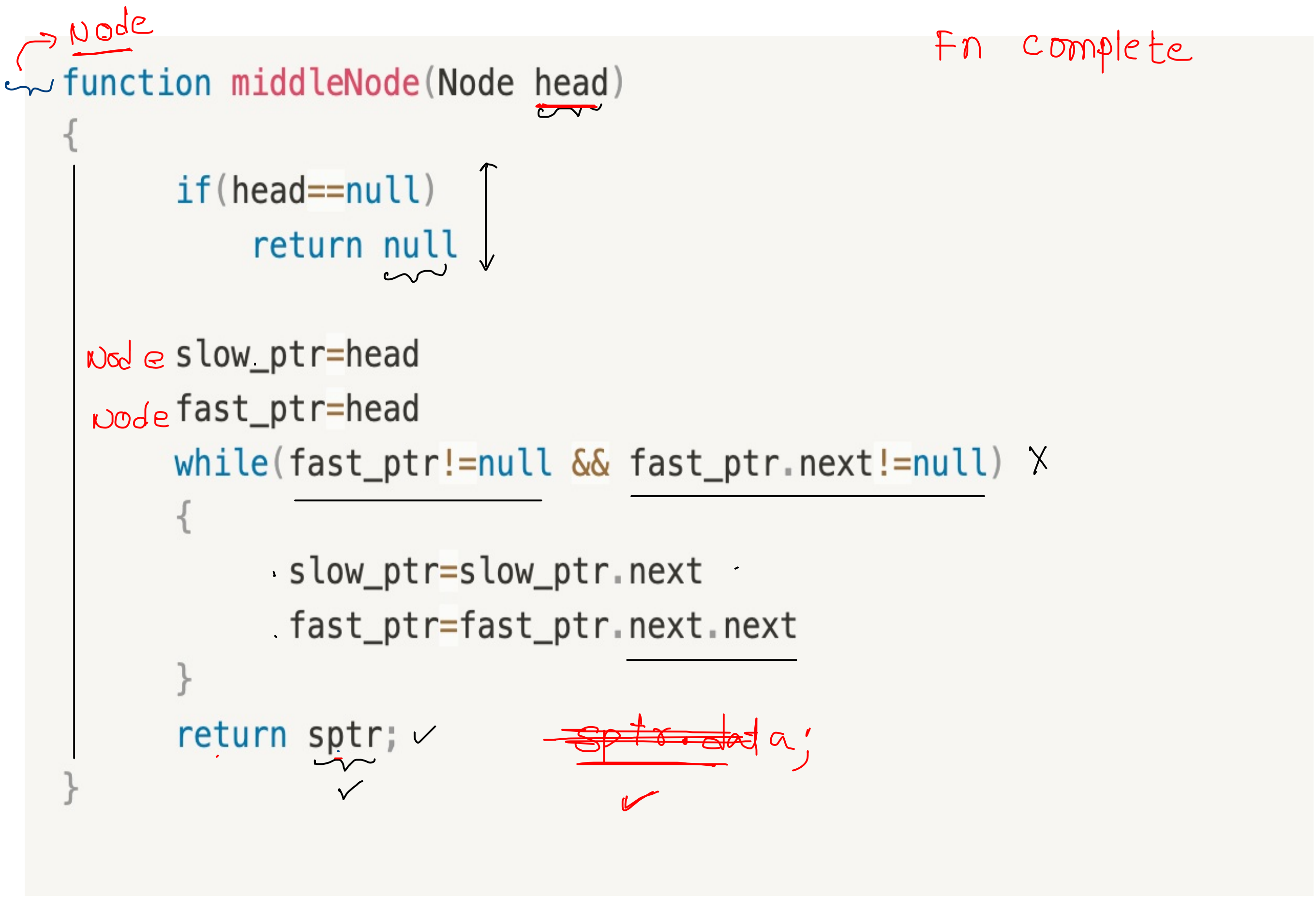
null.something

Access

C1

f != null





Node

Fn complete

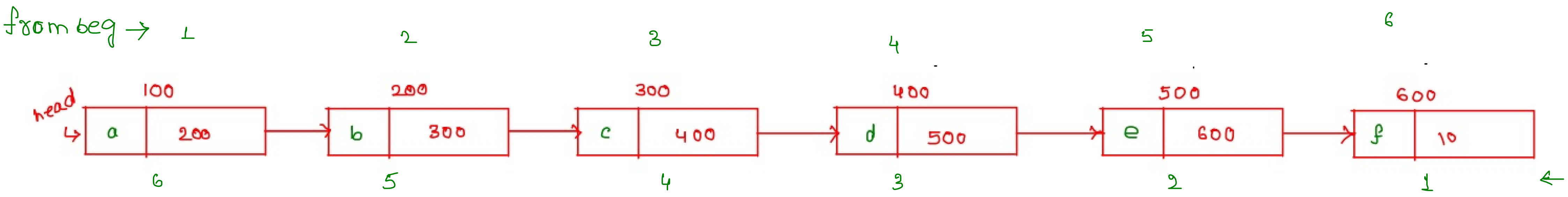
```
function middleNode(Node head)
{
    if(head==null)
        return null

Node slow_ptr=head
Node fast_ptr=head
    while(fast_ptr!=null && fast_ptr.next!=null)  X
    {
        slow_ptr=slow_ptr.next
        fast_ptr=fast_ptr.next.next
    }
    return sptr; ✓
}
```

~~sptr.data~~ a; ✓

3) Kth Node from the End [Accolite, Adobe, Amazon, FactSet, Hike, MAQ Software, Qualcomm, Snapdeal]

n = 6 ✓



i/p :- head, K = 4 o/p :- C

n = 6

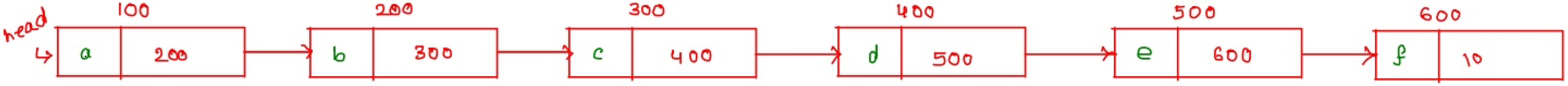
end		beg	n − K + 1
K = 1	→	6	(6 − 1 + 1)
K = 2	→	5	(6 − 2 + 1)
K = 3	→	4	(6 − 3 + 1)
K = 4	→	3	
K = 5	→	2	
K = 6	→	1	

1) Find length : n

2) from beg : n − K + 1

✓ observation:-

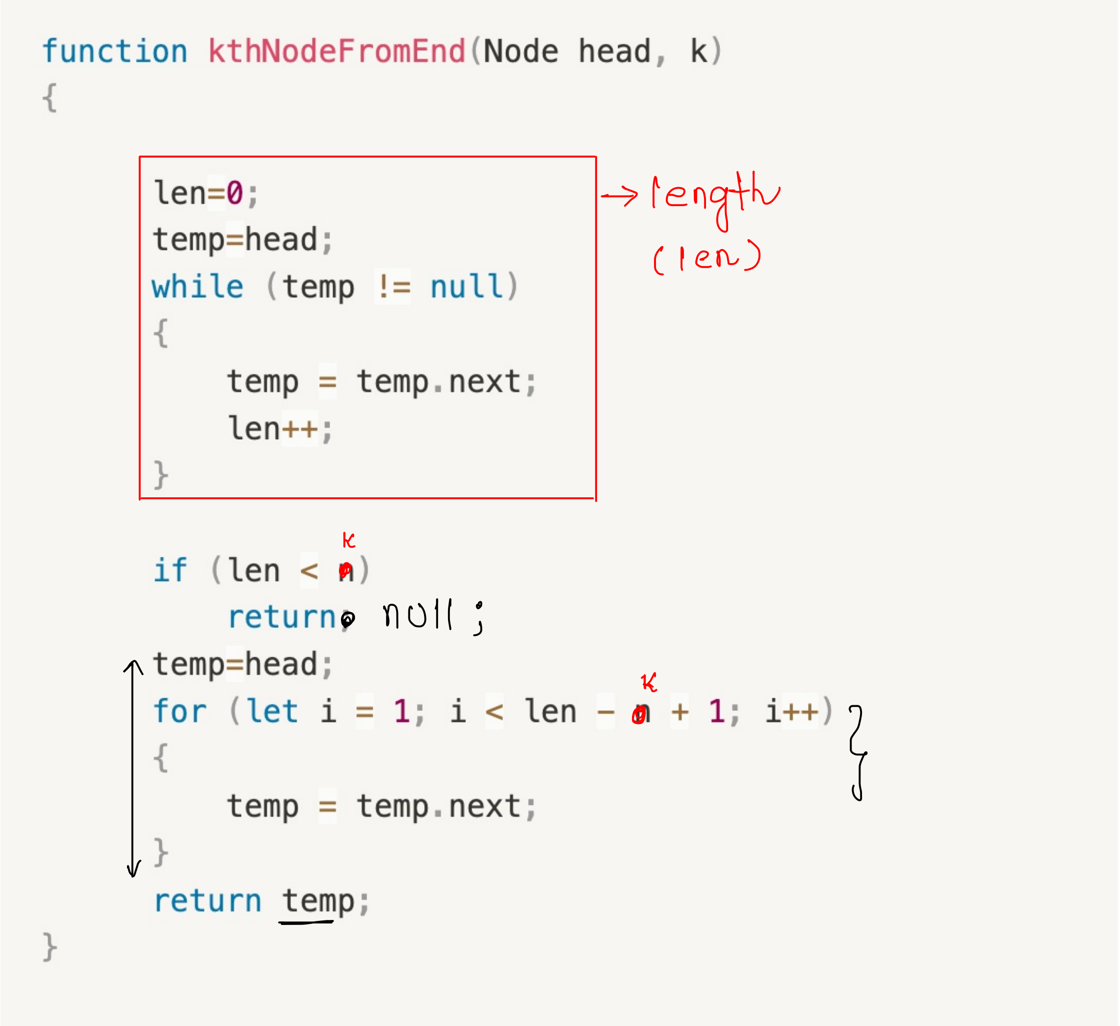
Kth Node from ending == (n-k+1) node from the begnning



len=6

7th node ending

6 Kids

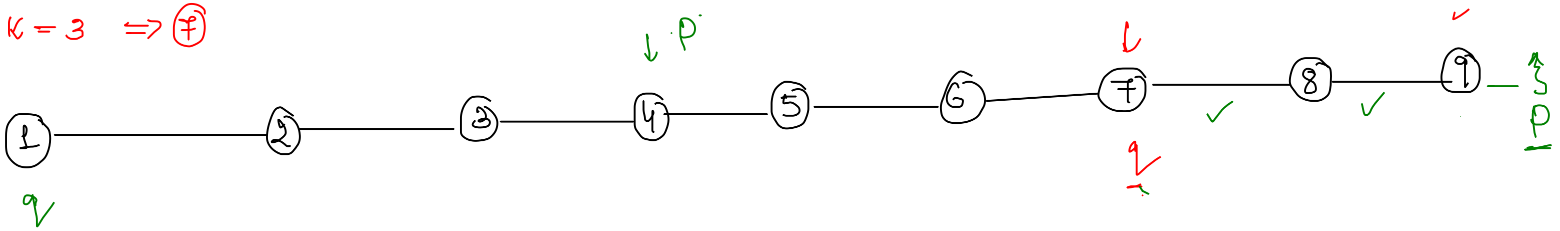


```
function kthNodeFromEnd(Node head, k)
{

    len=0;
    temp=head;
    while (temp != null)
    {
        temp = temp.next;
        len++;
    }

    if (len < k)
        return null;
    temp=head;
    for (let i = 1; i < len - k + 1; i++)
    {
        temp = temp.next;
    }
    return temp;
}
```

→ length (len)



p = q = head

steps:-

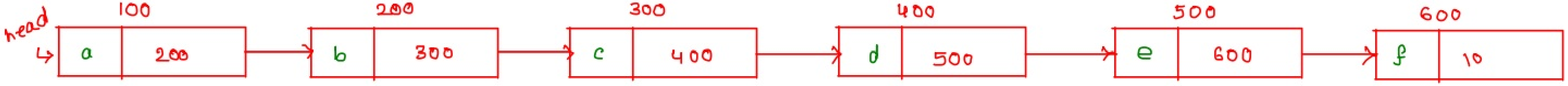
Loop(k) → ① place one ptr (p) @ beg of kth node

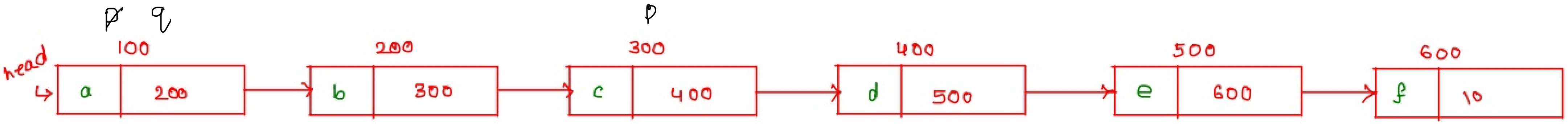
② move both ptrs @ same speed

```
while(p.next != NULL)
{   p = p.next
    q = q.next
}
```

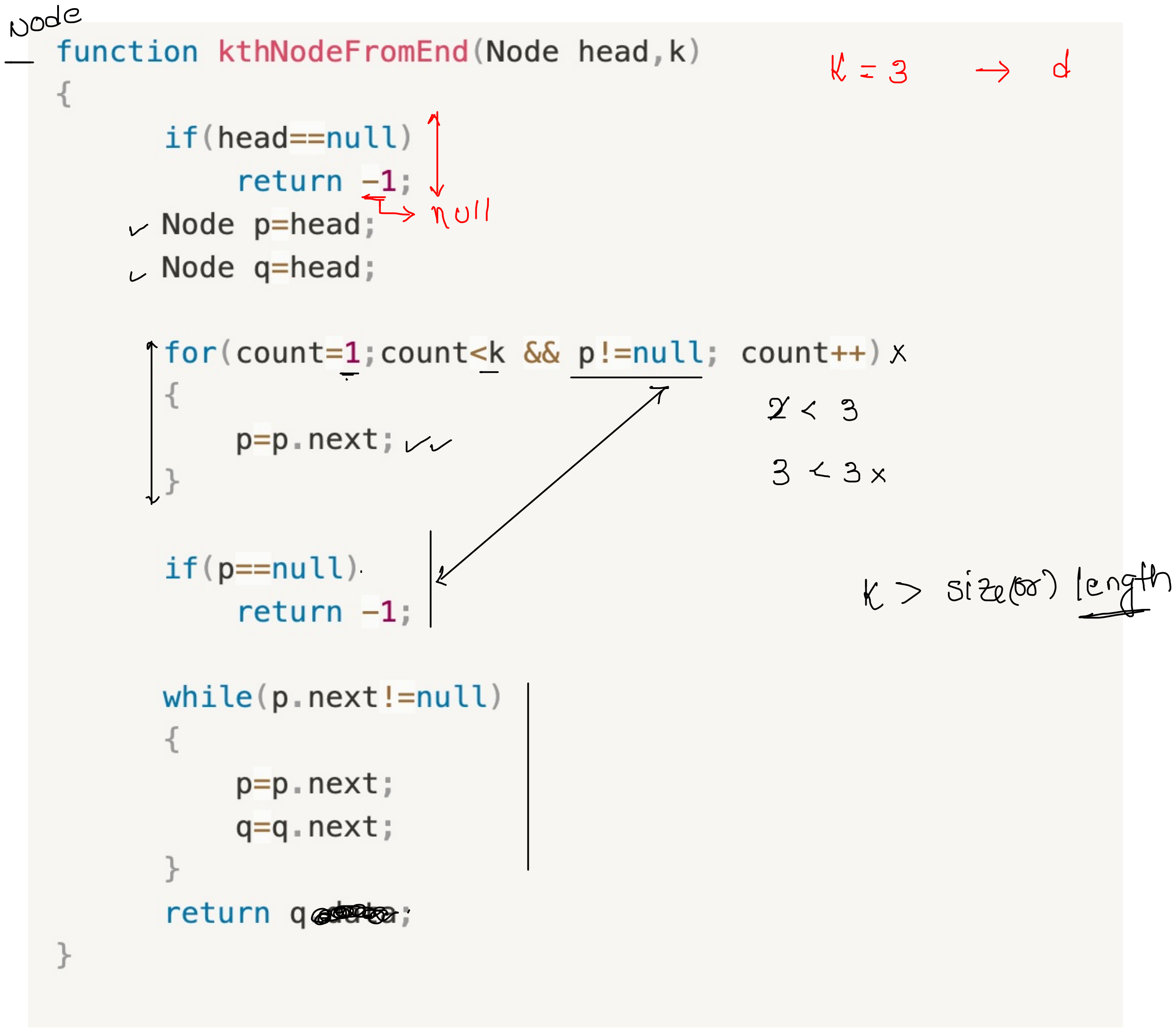
③ return q

Approach2:- Without Using Length





Node



```
function kthNodeFromEnd(Node head,k)
{
      if(head==null)
          return -1;
    ✓ Node p=head;
    ✓ Node q=head;

      for(count=1;count<k && p!=null; count++) x
      {
          p=p.next; ✓✓
      }

      if(p==null)
          return -1;

      while(p.next!=null)
      {
          p=p.next;
          q=q.next;
      }
      return q~~.data~~;
}
```

null

k = 3 → d

2 < 3

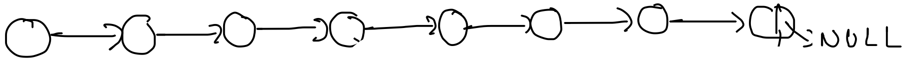
3 < 3 x

k > size(or) length

12 : 38pm

5) Detect loop / cycle in a Linked List [Accolite, Amazon, Samsung, MAQ Software]

↳ Floyd cycle detection Algo

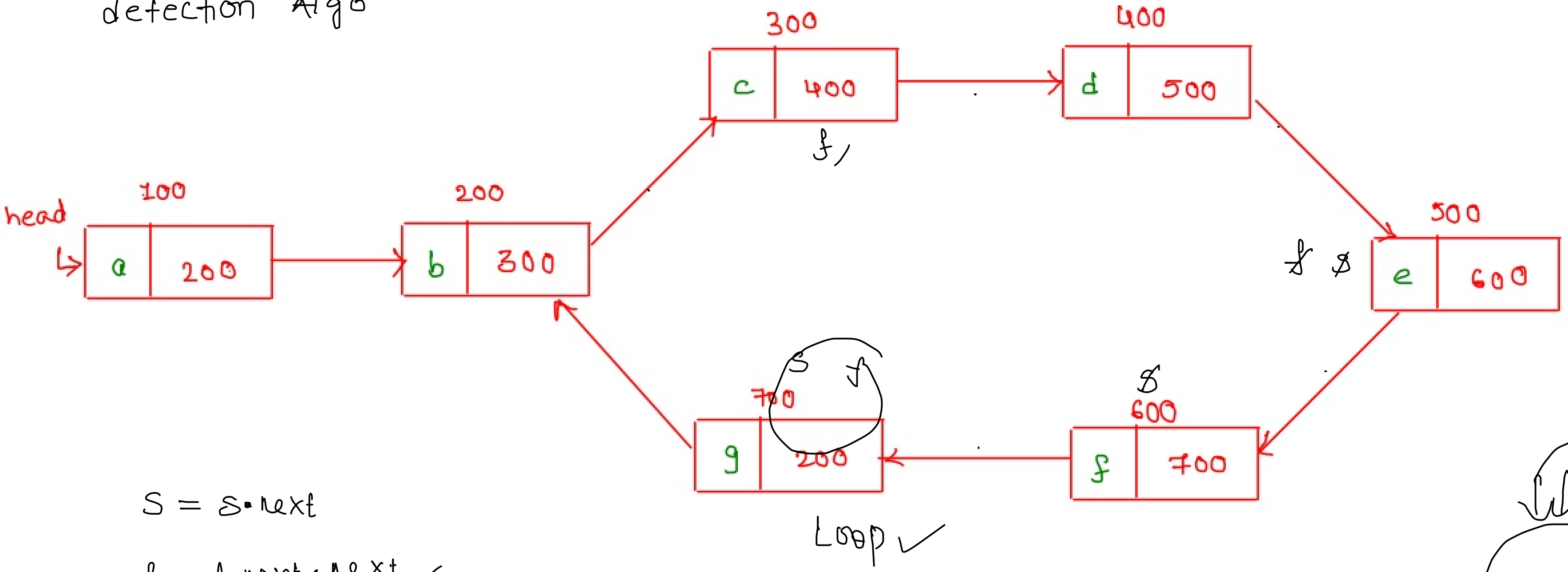


i/p :-

Loop : T

No ~~loop~~ loop : F

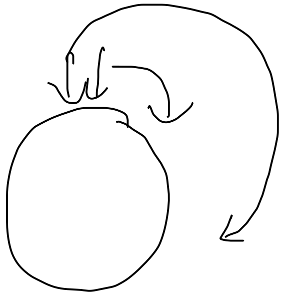
Ex:-



→ T

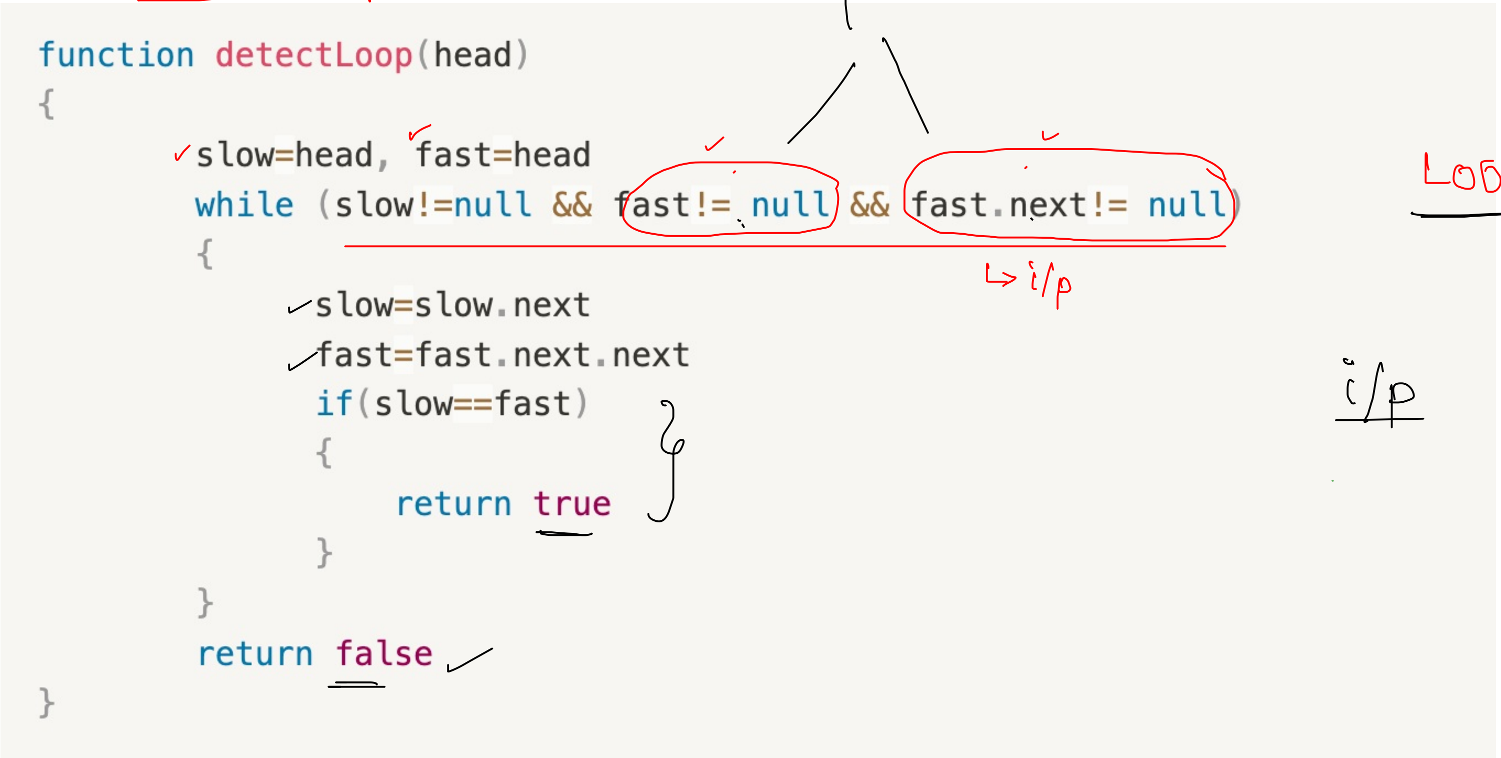
S = s.next

f = f.next.next ✓



FCD Algo

fast ! =NULL

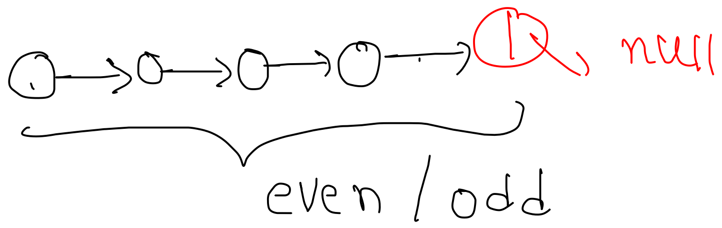


```
function detectLoop(head)
{
        slow=head, fast=head
        while (slow!=null && fast!= null && fast.next!= null)
        {
            slow=slow.next
            fast=fast.next.next
            if(slow==fast)
            {
                return true
            }
        }
        return false
}
```

↳ i/p

Loop → not there

i/p


null
even / odd