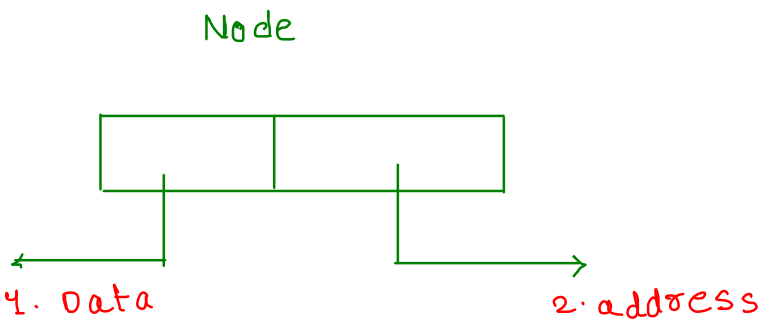
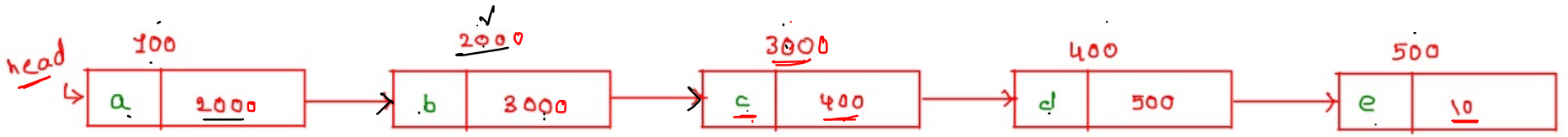
Master Class - 4 [Linked List]

- Venu Gopal

A linked list is a linear data structure consisting of a group of nodes where each node points to the next node through a pointer. Each node is composed of data and a reference (in other words, a link) to the next node in the sequence.

```
Class Node
{
        data
        next
}
```





NULL
(\0)
↳ not 40

a, b, c, d, e
data

SLL

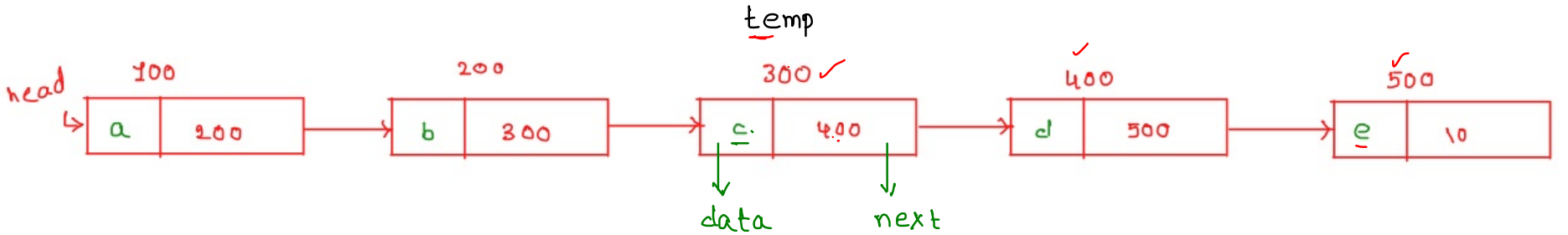
Note:-

r -> variable

-> Head pointer will always points to begnning of the list

100, 200, 300, 400, 500 (Assumptions)
addresses

-> Last node address is always NULL in a Single linked list



1) print(temp)

300

2) print(temp.data)

300 c

3) print(temp.next)

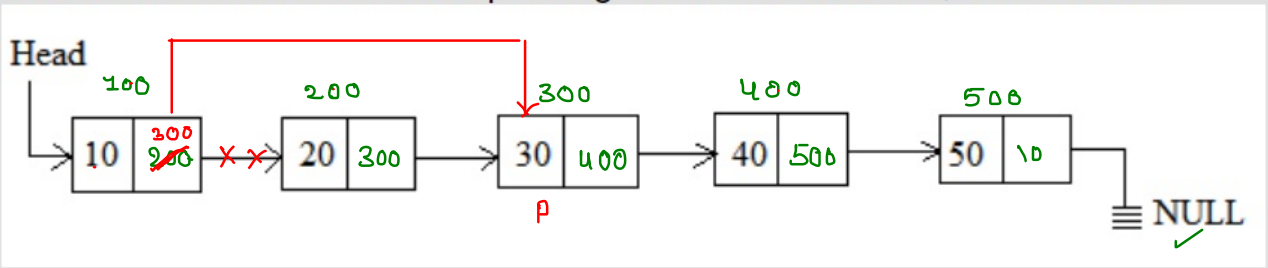
300 400

print(temp.next.next.data)

300 400 500 e

⇒ e

1. Given a linked list L with head pointing to the first node of L, shown below:



What is the output when the following sequence of operations applied on the given linked list?

P is a node pointer

(i) P = head → next → next; ⇒ p = 300

(ii) head → next = P;

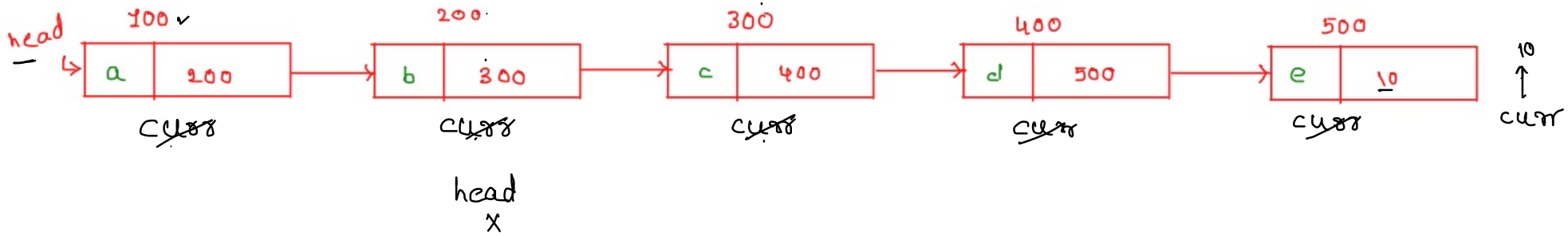
(iii) printf("%d", head → next → next→ data);

The output of the following code is 40 (Marks: 0.00)

LL ⇒ function complte.

1) Find the length of SLL

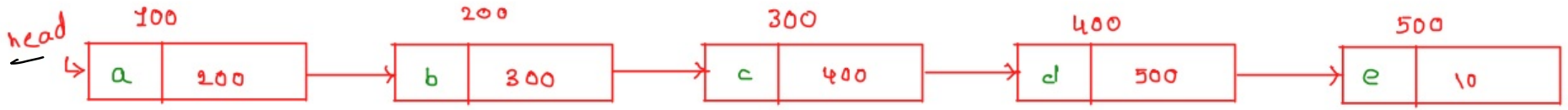
o/p 5



head
x

curr = head

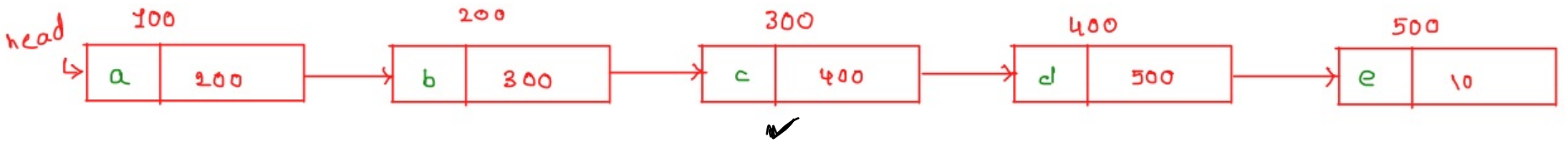
Count = 0 1 2 3 4 5 ✓




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

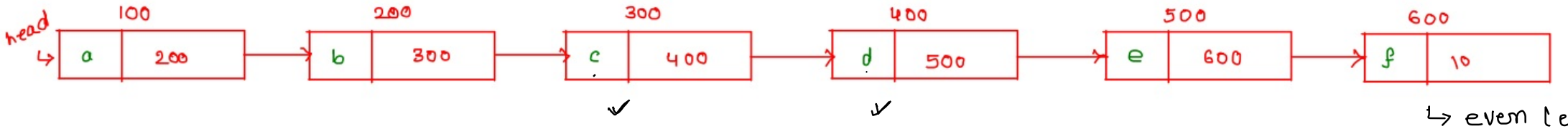
PS-code

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



→ odd length

c

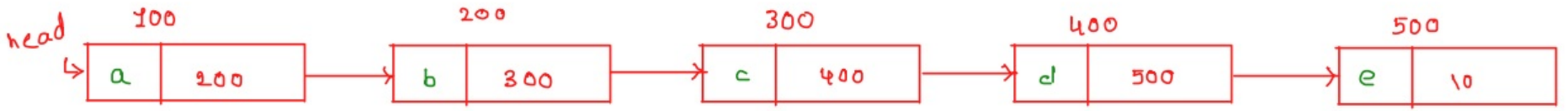


↳ even length

c

d

Ap1 :-

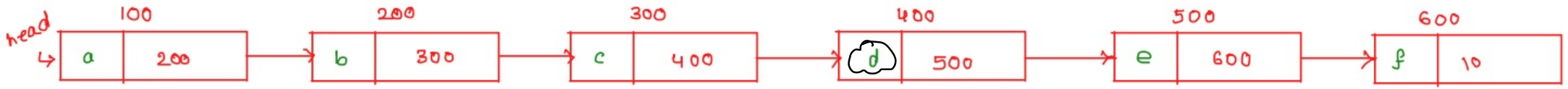


L=5

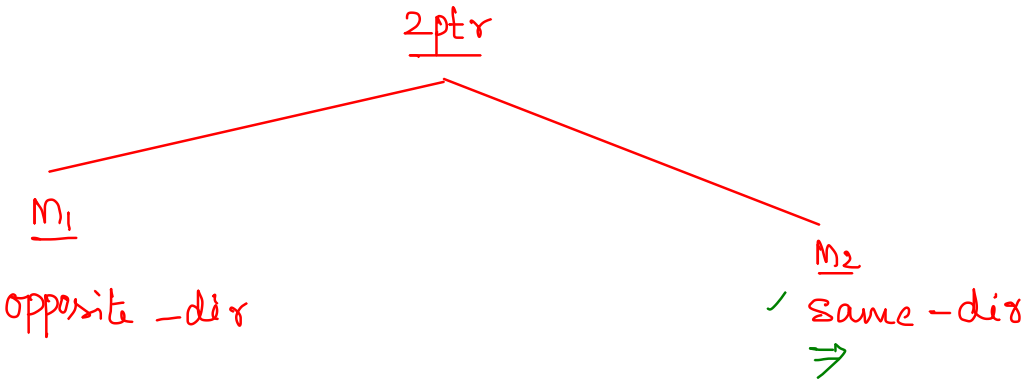
$\lceil \frac{5}{2} \rceil = 3$

len ⇒ return (len/2) element

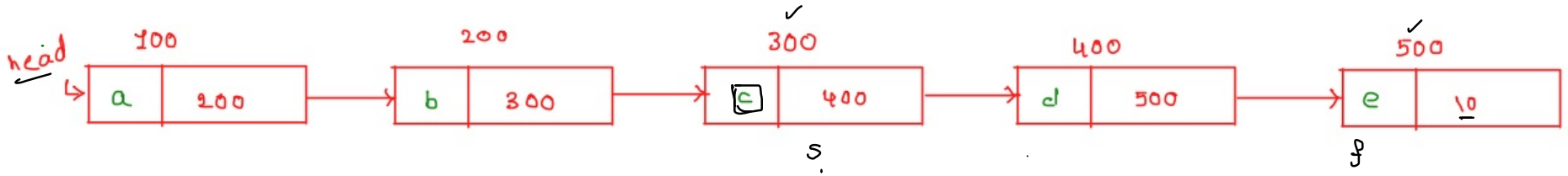
L=6



$\lceil \frac{6}{2} \rceil + 1$


2ptr
M1
opposite -dir
M2
same -dir

AP2 (2ptr)



s → moves by 1 step

f → moves by 2 steps

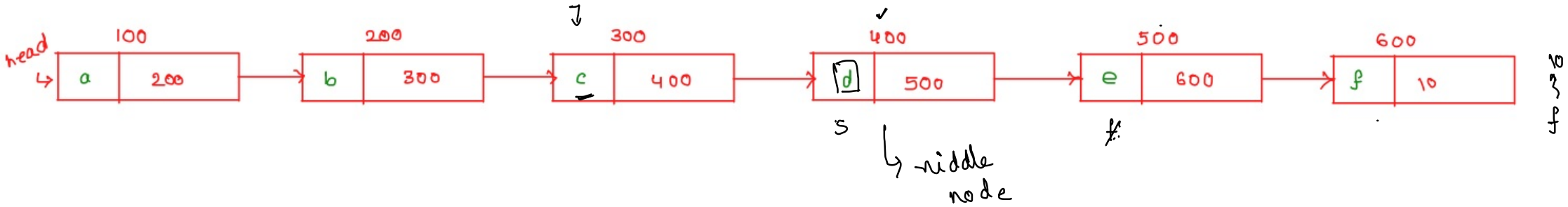
```
function middleNode(Node head)
{
    *if(head==null)
        return null

    1.slow_ptr=head
    2.fast_ptr=head
  ↳ while(fast_ptr!=null && fast_ptr.next!=null)
    {
        1.slow_ptr=slow_ptr.next
        2.fast_ptr=fast_ptr.next.next
    }
    return sptr;   slow_ptr;
}
```

Stop

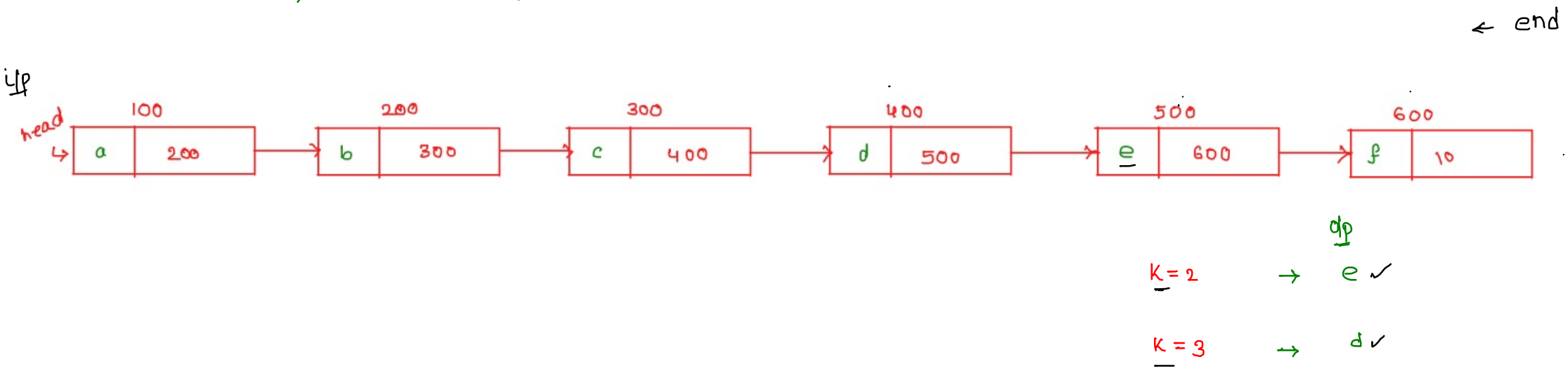
x

c1 c2

40 → 1 → 9

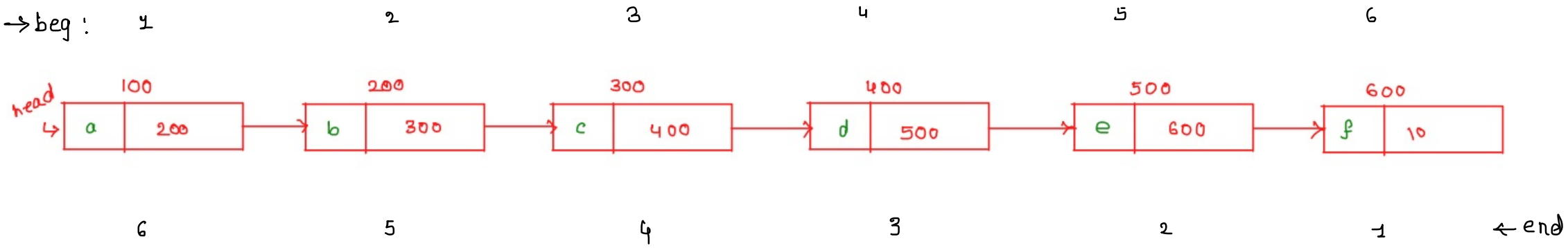

head
100
a
200
200
b
300
300
c
400
400
d
500
500
e
600
600
f
10
s
f
middle node

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

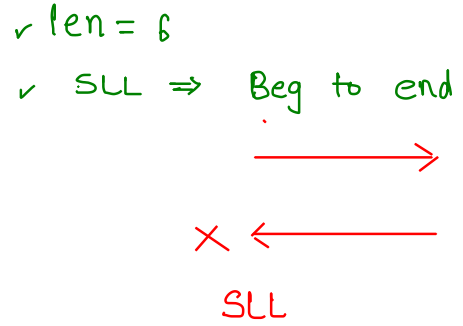


Approach1 :- Using Length

k = 2

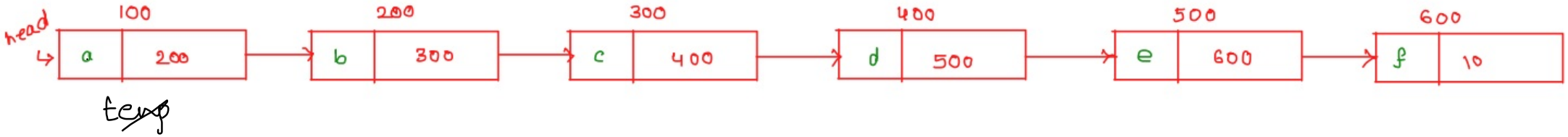


from ending	from beginning	len = 6
k = 2	5th from beg	6 − 2 + 1
k = 3	4th from beg	6 − 3 + 1
k = 4	3rd "	6 − 4 + 1



observation:-

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




```
function kthNodeFromEnd(Node head, k)
{
    len=0;
    temp=head;
    while (temp != null)
    {
        temp = temp.next;
        len++;
    }

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

k=3

len=5

k=5

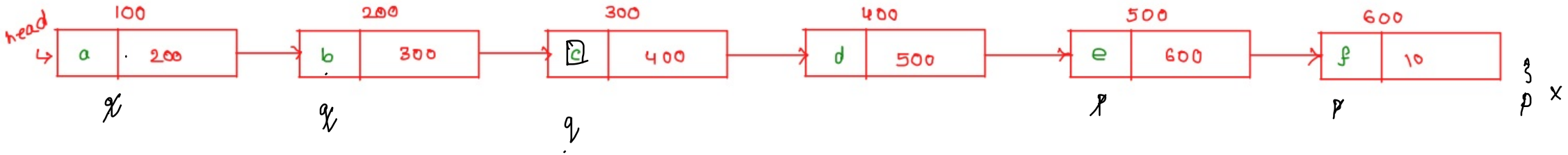
len ≥ k ✓

(temp.data)

Approach2:- Without Using Length and without reversing (2ptr -tech)

k=2 ⇒ e

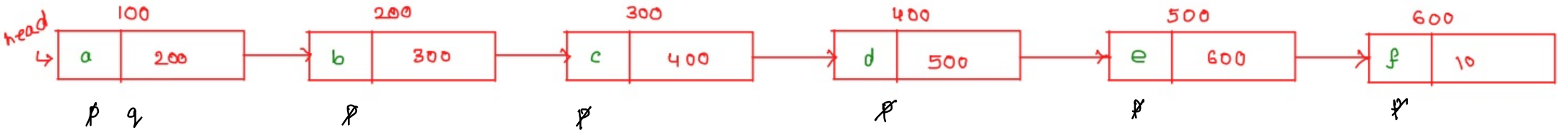
k=4 ⇒ c



1. P, Q ⇒ head

2. (P) ⇒ after k nodes from beginning

3.



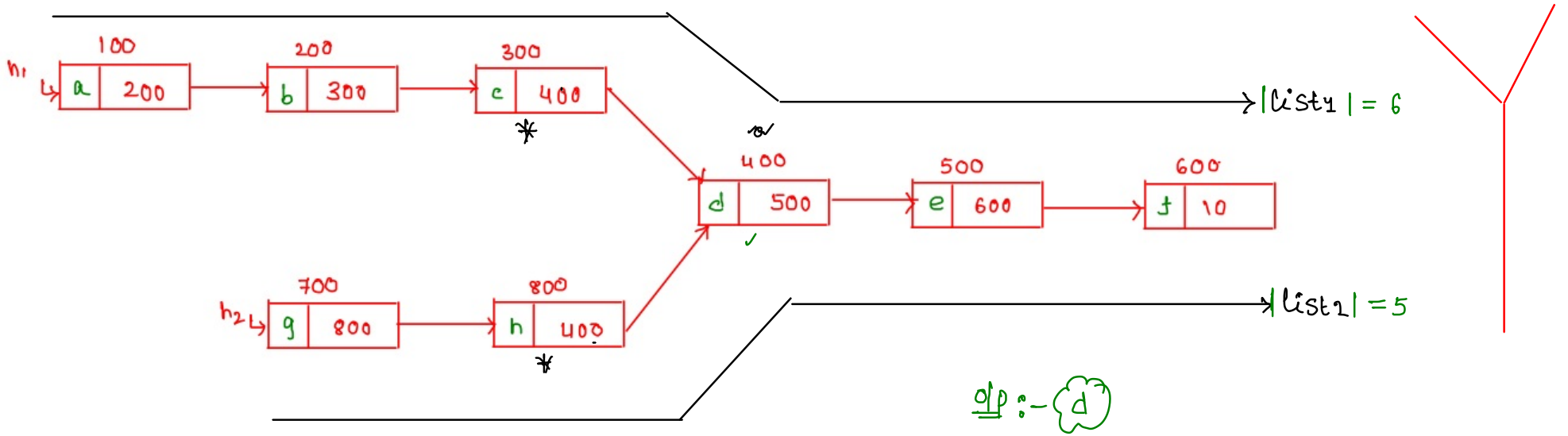
k > len ⇒ stop

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

      for(count=1;count<k && p!=null; count++)
      {
          p=p.next;
      }
      if(p==null)
          return -1;

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

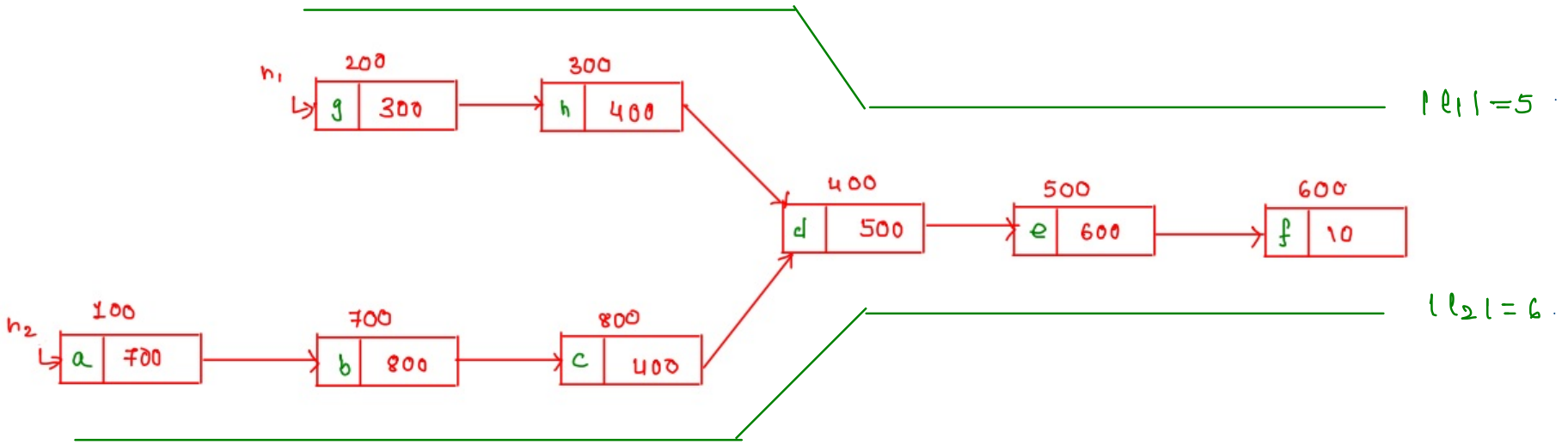
k = 7

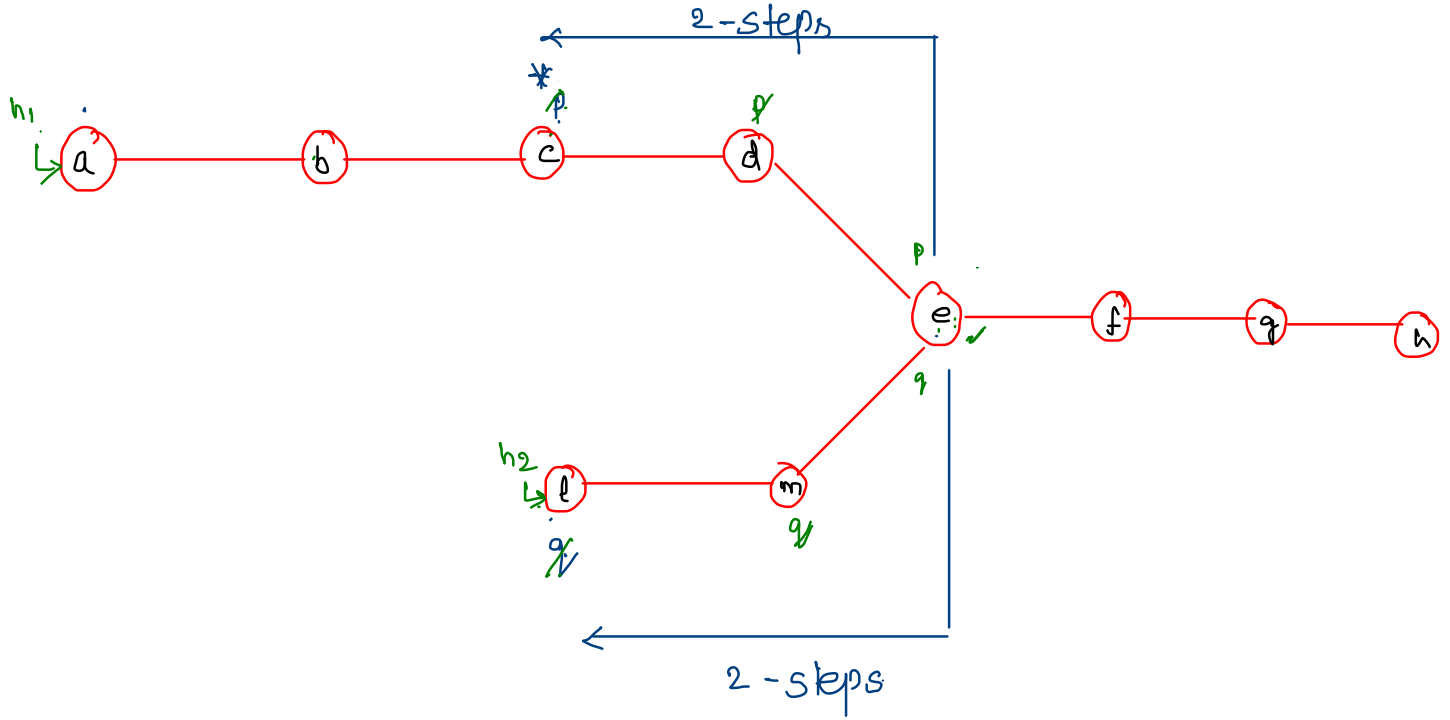
count = 1 2 3 4 5 6 7

4) Find the Intersection node in Y shaped linked list [Accolite, Amazon, D-E-Shaw, Goldman Sachs, MakeMyTrip, Microsoft, Qualcomm, Visa, Zopper]



op :- d


h1
200
g
300
300
h
400
400
d
500
500
e
600
600
f
10
h2
100
a
700
700
b
800
800
c
400
|l1|=5
|l2|=6



(e)

1. len(h1) = 8 ✓

2. len(h2) = 6 ✓

3. diff = 2 ✓

|8 − 6| =

```
function getIntersectionNode(head1, head2)
{
    l1=length(head1)  → 6
    l2=length(head2)  → 5
    if(l1==0 || l2==0)
        return -1;
    if(l1>l2)
    {
        diff=l1-l2
        return fun(diff, head1, head2);
    }
    else
    {   diff=l2-l1
        return fun(diff,head2,head1)
    }

}
```

$l_1=6 \Rightarrow$

$l_2=5$

l1 is big l2

l2 is big l1

```
function fun(diff,head1,head2)
{
    curr1=head1;
    curr2=head2;
    for(i=1;i<=d;i++)
    {
        if(curr1==null)
            return -1;
        curr1=curr1.next;
    }
    while(curr1!=null && curr2!=null)
    {
        if(curr1==curr2)
            return curr1.data
    }
    return -1;
}
```

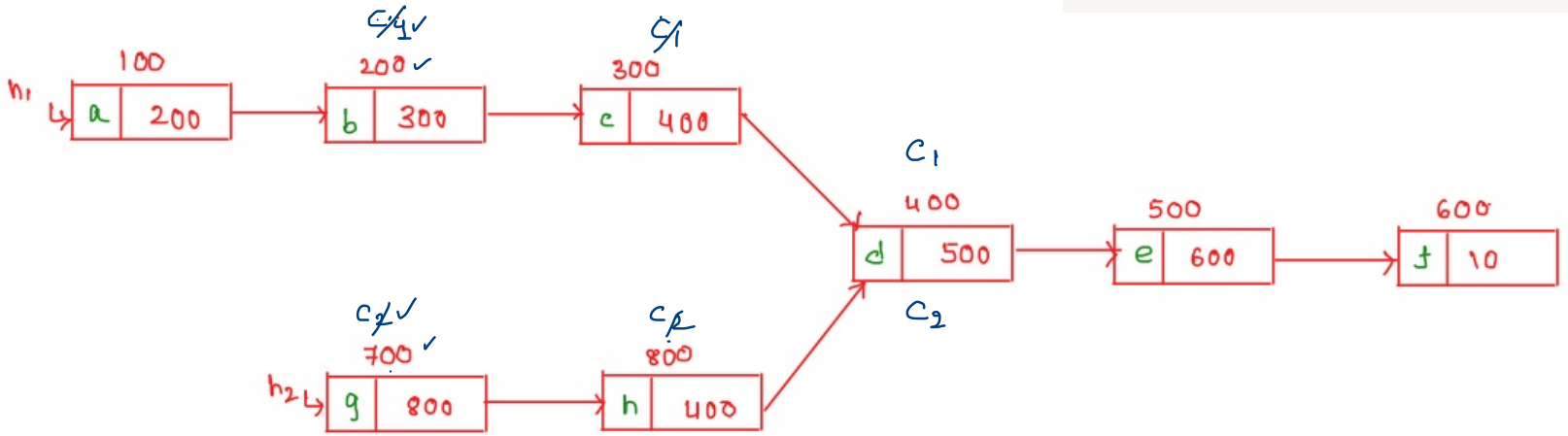
d → largest list

d = 1

i = 1 2

curr1

curr1 = curr1.next ; curr2 = curr2.next ;



11:18

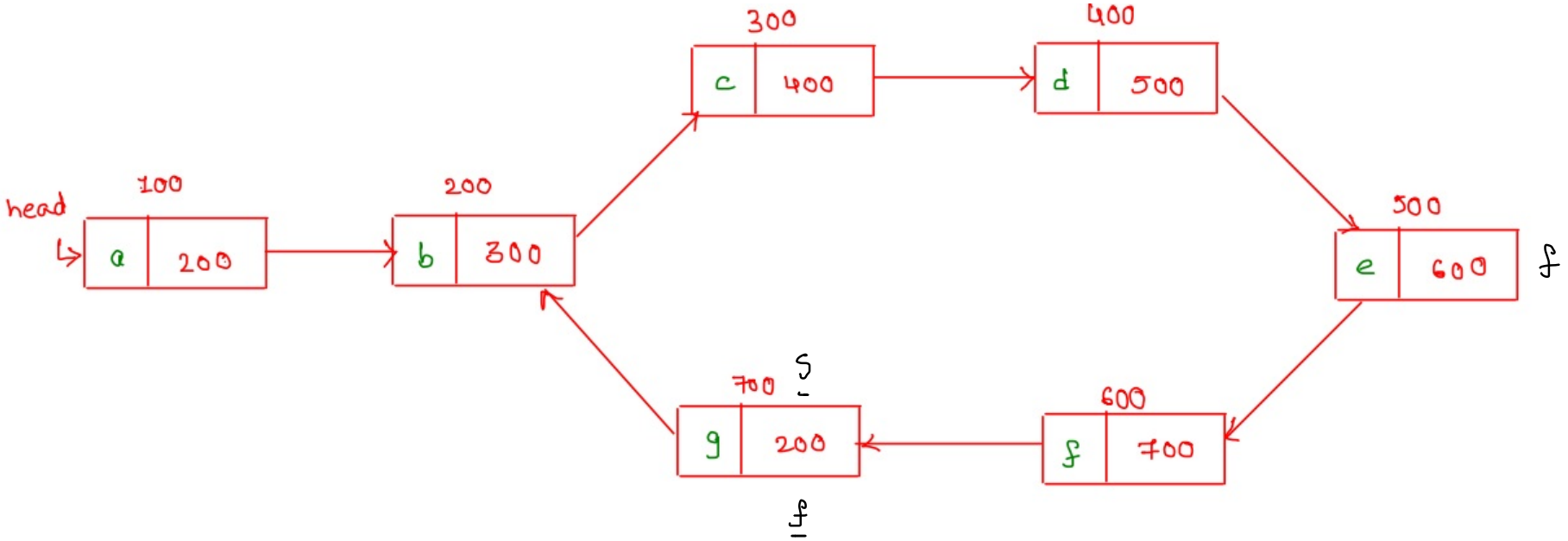
5) Detect loop / cycle in a Linked List [Accolite, Amazon, Samsung, MAQ Software]

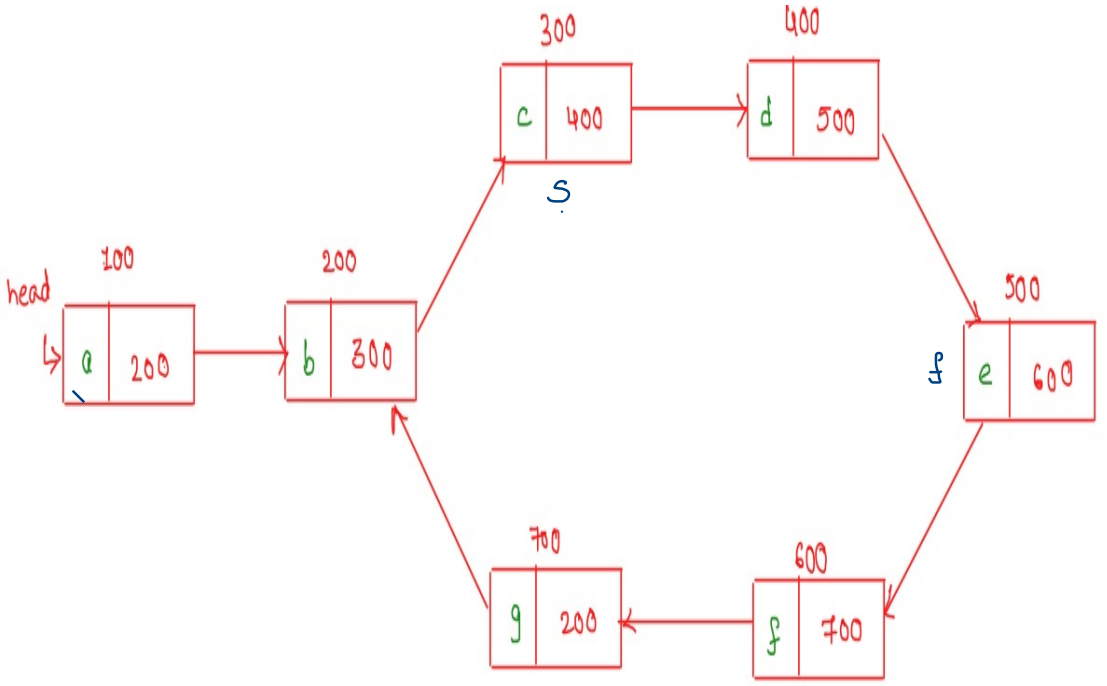
↳ Floyd cycle detection Algo ⇒ 2ptr

· Loop → True ·

· No Loop → False

↳ last nod add → NULL





No-loop ⇒ 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
}
```

↳loop

↳no loop

null.next ↳

if ↳loop ✓ → s, f
↳loop ✗ ↘
s, f

s → s.next ✓

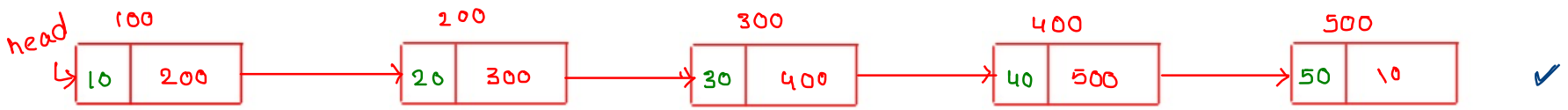
f → f.next.next

1. ✓ s, f ✓

2. ✓ ⇒ if
↳No-loop
↳ f = f.next.next

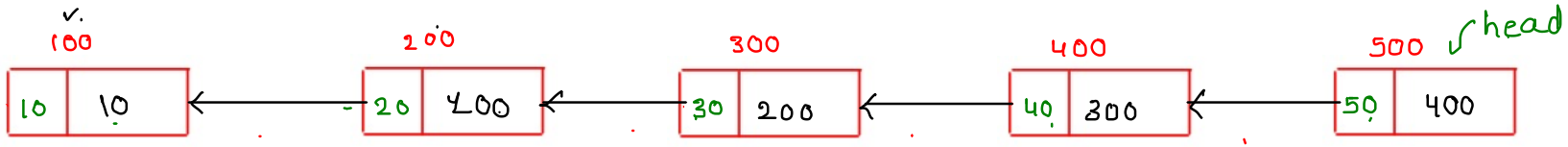
↳ ⇒odd

6) Reverse a SLL [Accolite, Adobe, Amazon, MakeMyTrip, Qualcomm, Samsung, SAP Labs, Snapdeal, Zoho]



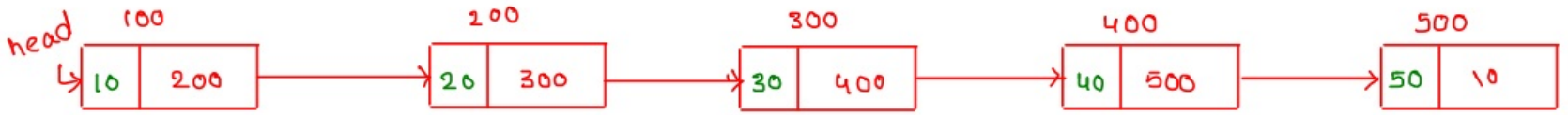
i/p :- 10 → 20 → 30 → 40 → 50

o/p :- 50 → 40 → 30 → 20 → 10 ✓



.prev next

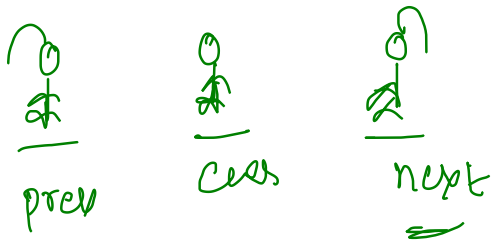
curr ⇒ null (Stop)



curr=head

prev =null

next =null

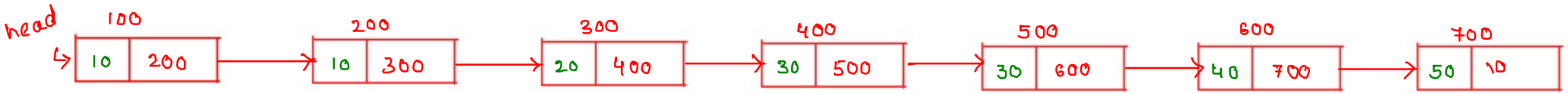



```
while(curr != null)
{
  1. next = curr.next
  2. curr.next = prev
  3. prev = curr
  4. curr = next
}
head = prev
```

```
function reverseSLL(head)
{
    1. prev=null
    2. curr=head
    3. next=null
      while(curr!=null)
      {
          1. next=curr.next
          2. curr.next=prev
          3. prev=curr
          4. curr=next
      }
      head=prev
      return head
}
```

7) Remove duplicates from sorted list [Adobe, Myntra, Visa]

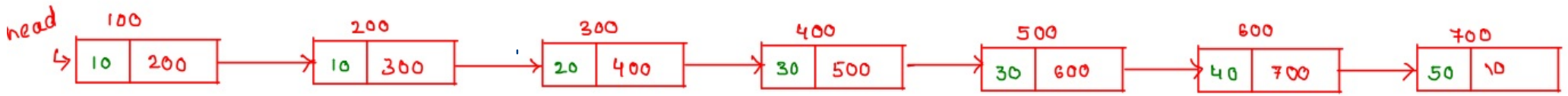
Sorted

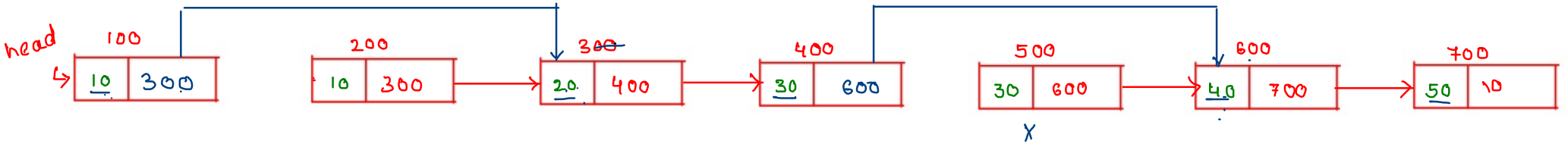


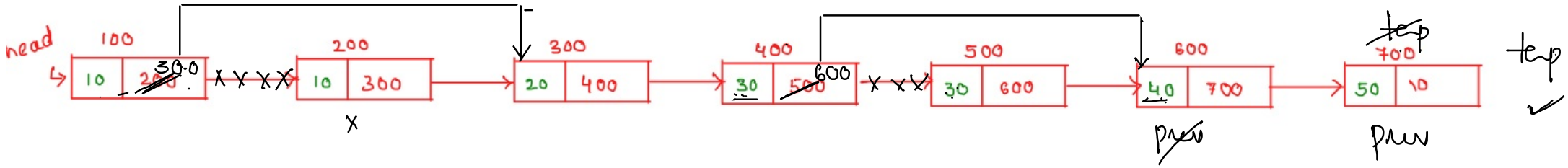
i/p :- 10 → 10 → 20 → 20 → 30 → 40 → 50

o/p :- 10 → 20 → 30 → 40 → 50 ⇒ ONLY distinct

play with add'ls.







10 → 20 → 30 → 40 → 50 (o/p)

temp.data V/S prev.data

==

no-need

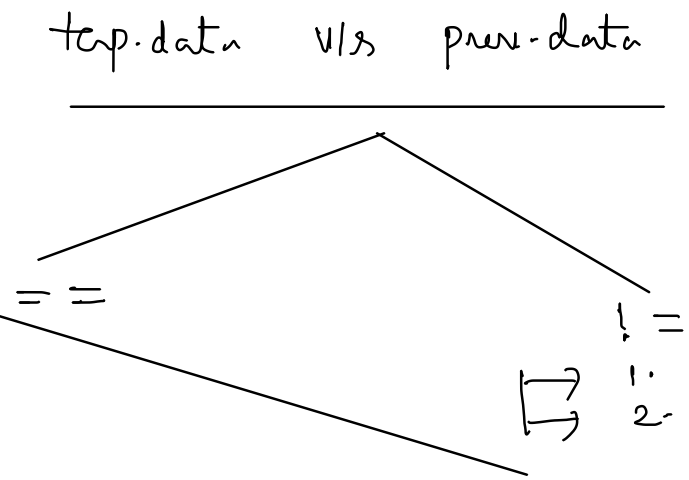
!=

1. prev.next = temp

2. prev = temp

```
function removeDuplicates(head)
{
     temp=head, prev=head
      while(temp!=null)
      {
            if(temp.data!=prev.data)
            {
                prev.next=temp
                prev=temp
            }
            temp=temp.next
      }
      if(prev!=temp)
      {
          prev.next=null
      }
      return head
}
```

⇒ why (?)

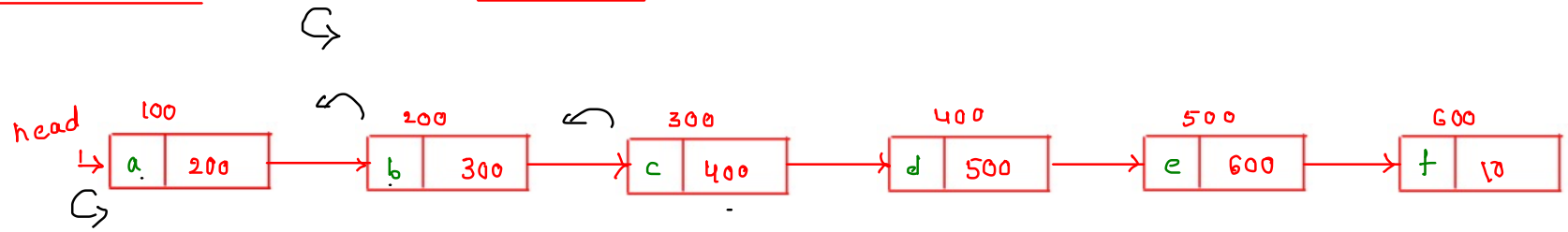


What if

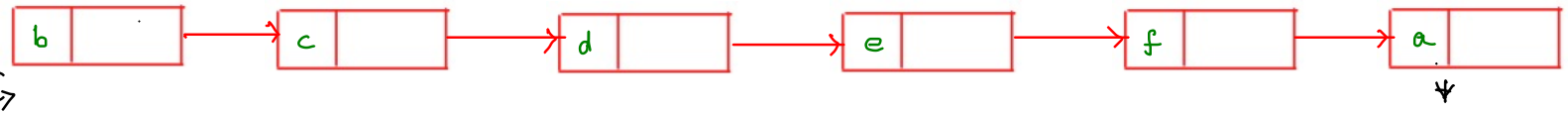
last two ele's are same

8) Rotate the SLL anti clock wise by k times [Amazon, Microsoft, Make My Trip, Accolite]

k = 4

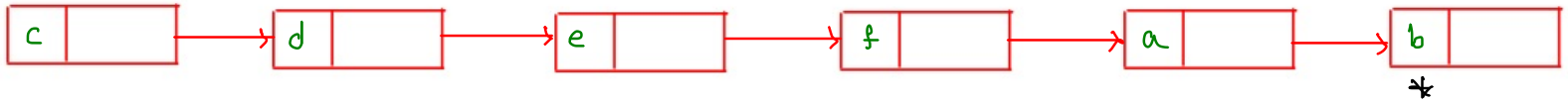


✓ k=1



✓

✓ k=2



✓ k=3

✓ k=4

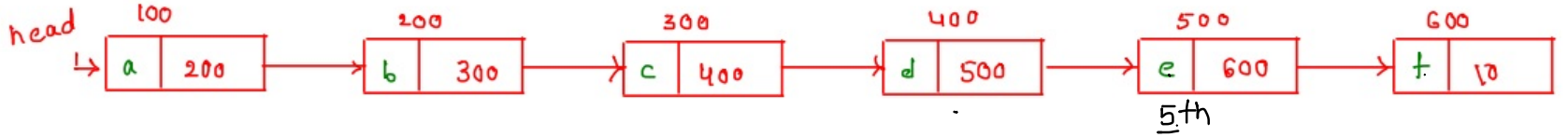
o/p ✓

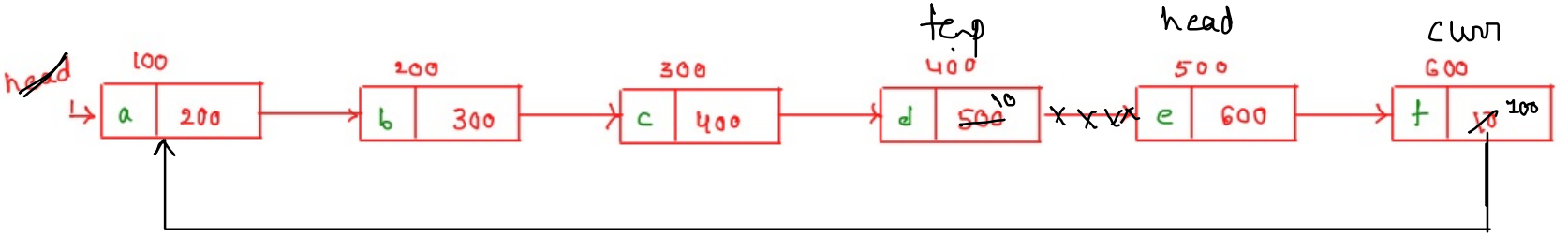
k > len len k = k % len

k = 4

↻ Anti-clock wise

i/p





1. temp → kth beg ✓

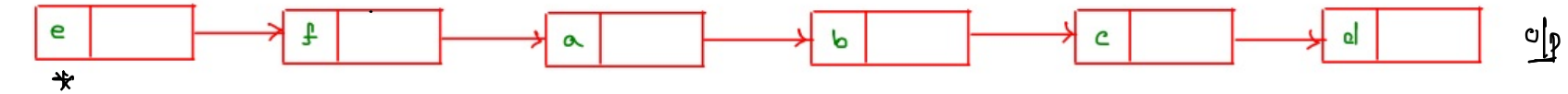
2. curr = temp.next

3. curr.next = head

4. head = temp.next

5. temp.next = null

o/p



o/p

```
function rotateSLL(head,k)
{
      if(k==0)
          return head;
      curr=head
      count=1
      while(curr<k && curr!=null)
      {
            curr=curr.next
            count++
      }
      if(curr==null)
          return head
      kthNode=curr
      while(curr.next!=null)
      {
          curr=curr.next
      }
      curr.next=head
      head=kthNode.next
      kthNode.next=null
}
```

9) Add two numbers represented by SLL [Accolite, Amazon, Flipkart, Snapdeal, Microsoft, Qualcomm]

9 → 9 → 3 → 7

n_1 = 9 9 3 7 (→ ←)

reverse →

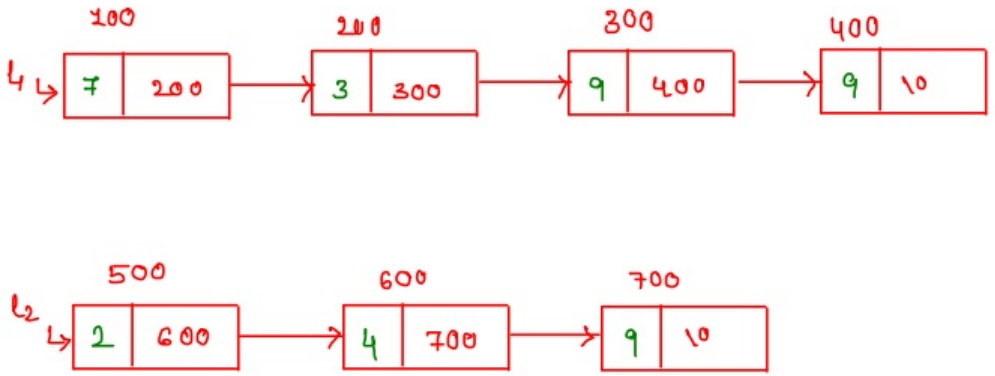
n_2 = 9 4 2

$n_1 + n_2$

1 9 9 3 7 ←

† 9 4 2

1 0 8 7 9 o/p



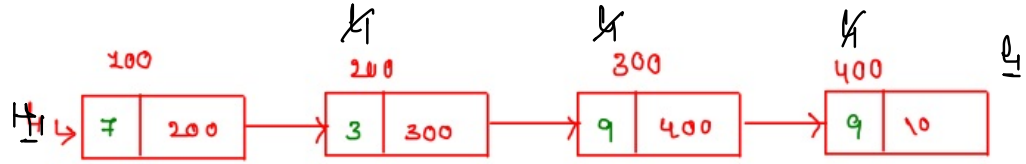
reverse_list

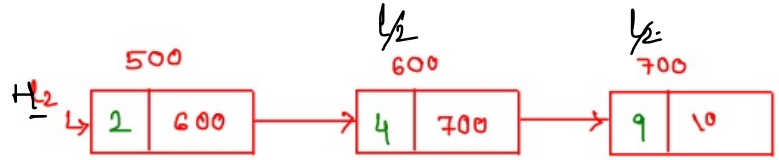
↓

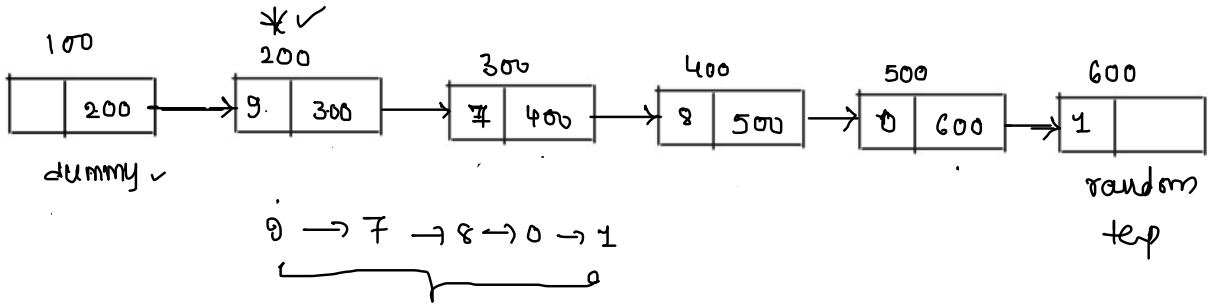
o/p reverse

```
function addTwoLinkedLists(l1,l2)
{

  dummy=null
  temp=dummy
  carry=0;
  while(l1!=null || l2!=null || carry==1)
  {
      sum=0
      if(l1!=null)
      {
        sum=sum+l1.data
        l1=l1.next
      }
      if(l2!=null)
      {
        sum=sum+l2.data
        l2=l2.next
      }
      sum=sum+carry
      carry=sum/10
      create a node  with the value (sum%10) [ random]
      temp.next=random
      temp=temp.next
  }
  return dummy.next
}
```

	1	1	0	0
		9	9	3	7
+			9	4	2
1	0	8	7	9







9 → 7 → 8 → 0 → 1

carry = 0 1 1 0

sum = 0 9 10 0 1

18/10 = 1

18 % 10 = 8

THANK-YOU