Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

# 876. Middle of the Linked List

Easy | Topics | Companies | Re-do

Given the `head` of a singly linked list, return *the middle node of the linked list*.

If there are two middle nodes, return **the second middle** node.

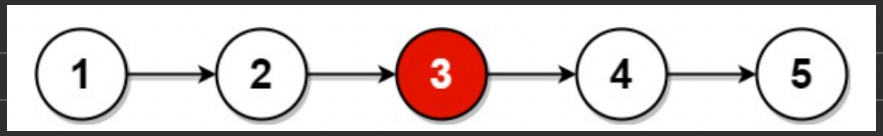

## ① find No. of nodes.

```
int noOfNodes ( Node head)
{
    Node temp = head
     cnt = 0
    while ( temp != null )
    {
        cnt++
        temp = temp . next
    }
      return cnt
}
```

```
Node middleNode(Node head)
{
    int n = noOfNodes(head) = O(N)
    int mid = (n/2)
    Node temp = head
    cnt = 0
    while (temp != null && cnt < mid)   } O(N/2)
    {
        temp = temp.next
        cnt++
    }
    return temp
}
```

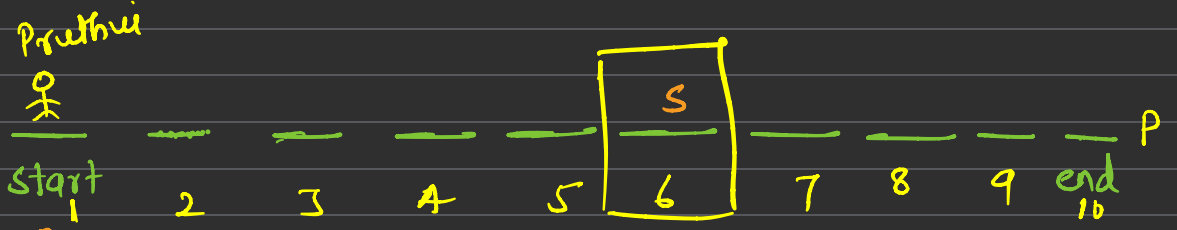
T.C = O(N) + O(N/2)

S.C = O(1)



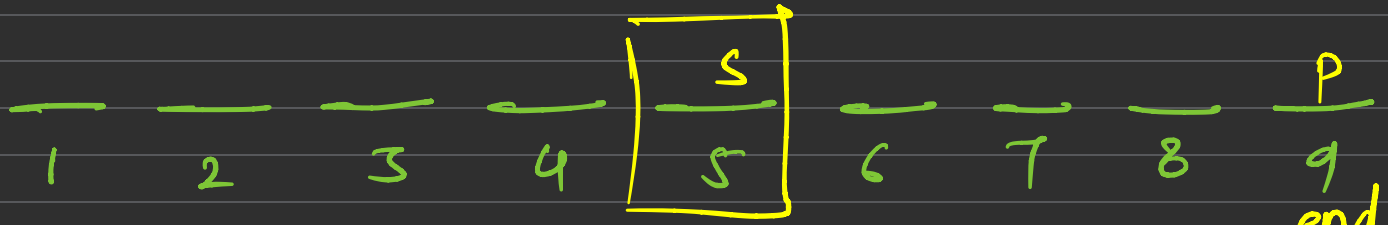

Even

N = 10

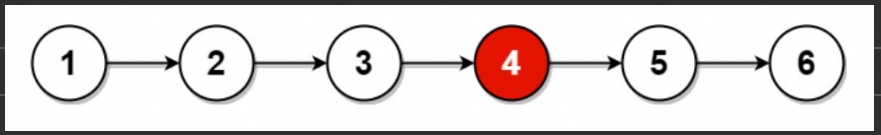



Prathui = 2 steps → fast

Sushtant = 1 step → slow




N = 9 odd




```
Node slow = head   fast = head

while ( fast != null && fast.next != null)
{
    fast = fast.next.next
    slow = slow.next

}

return slow
```

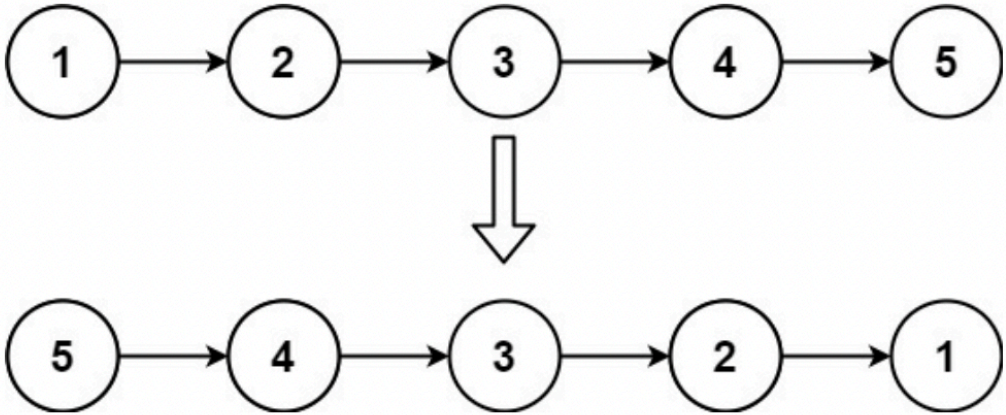
# 206. Reverse Linked List

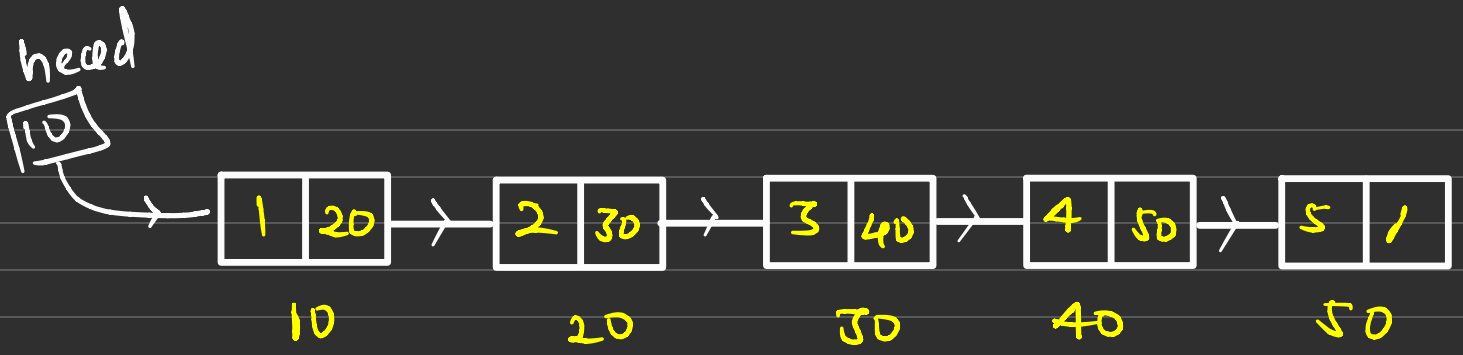
Solved

Easy | Topics | Companies | Re-do

Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

**Example 1:**



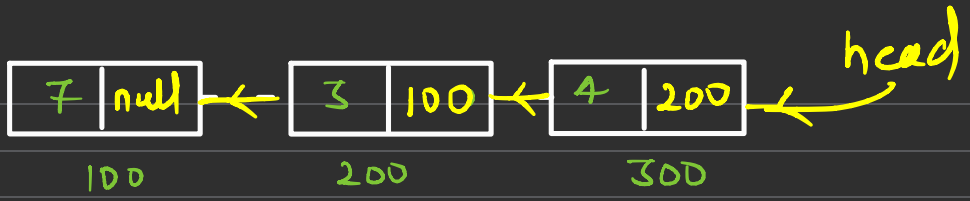


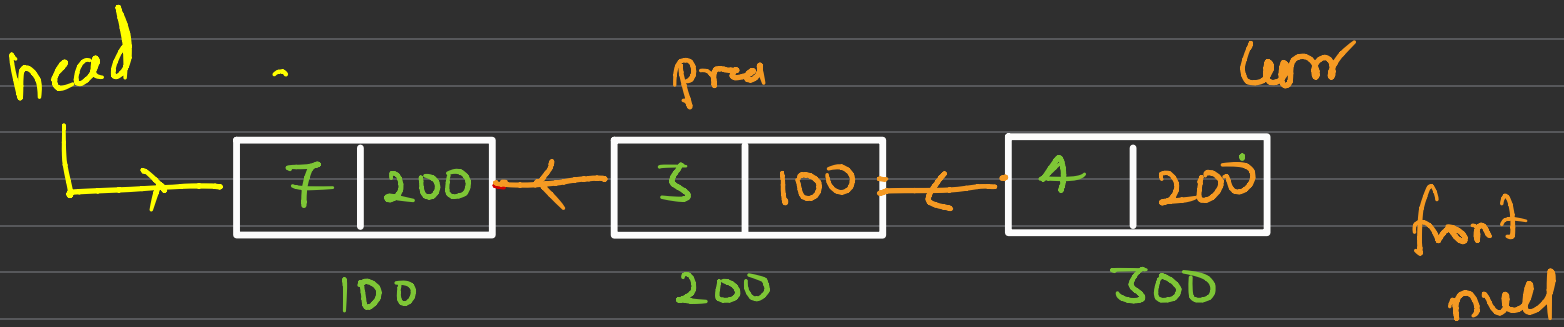

O(N)

1. Traverse Linked list & insert elements in array
2. Reverse the array O(N/2)
3. Put all elements from array to LL from head of list O(N)

$$T.C = O(N) + O(N/2) + O(N) = O(3N)$$

$$S.C = O(N)$$



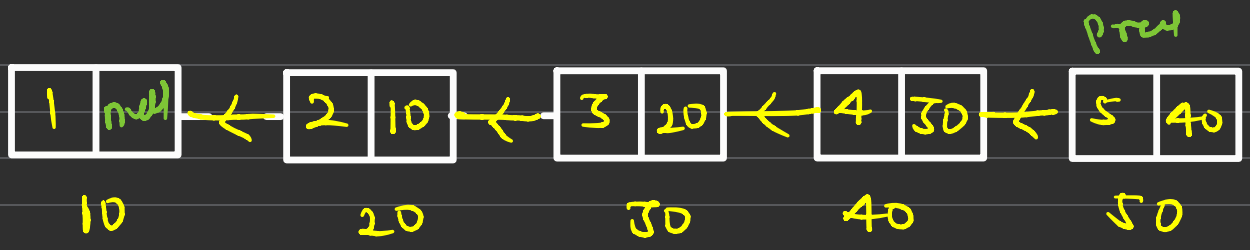




front = curr.next

curr.next = prev

prev = curr

curr = front



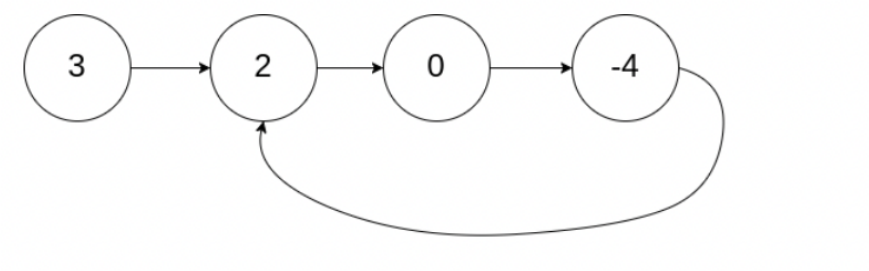

curr = head , prev = null , front = null

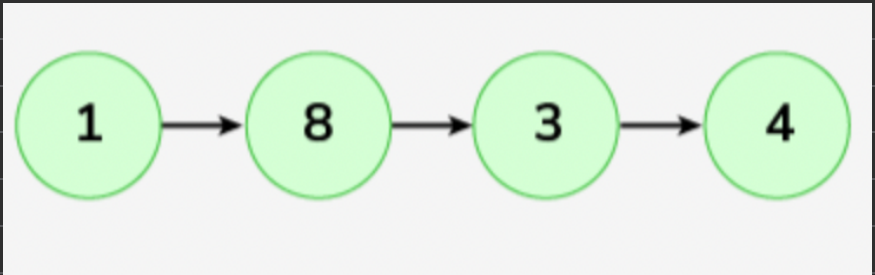
```
while ( curr != null)
{
    front = curr.next
    curr.next = prev
    prev = curr
    curr = front
}

return prev
```

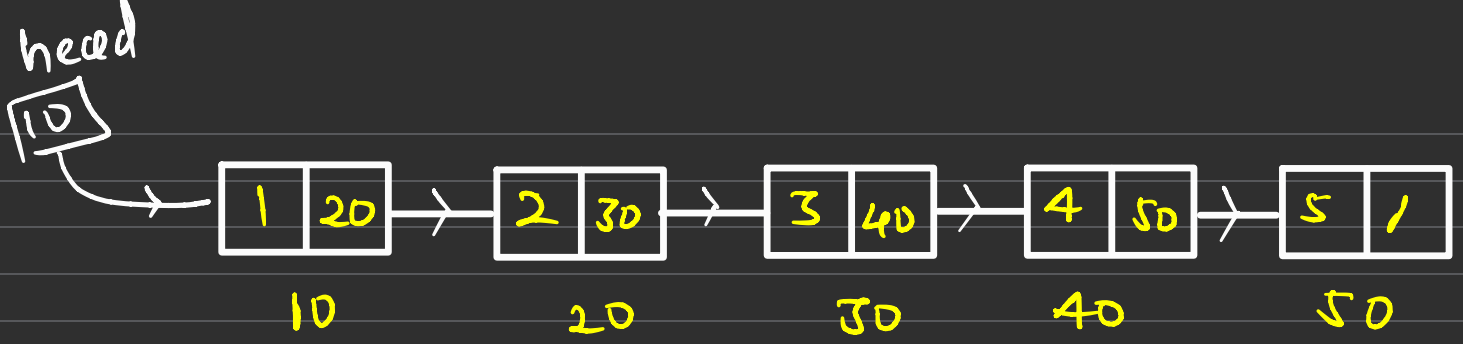
T.C = O(N)

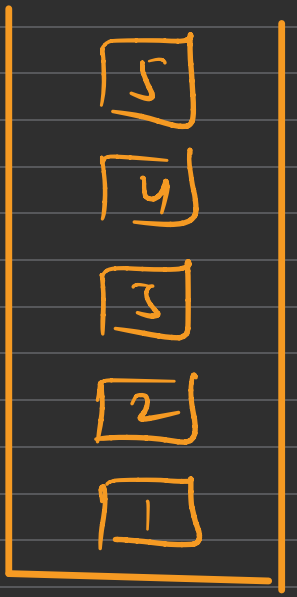
S.C = O(1)

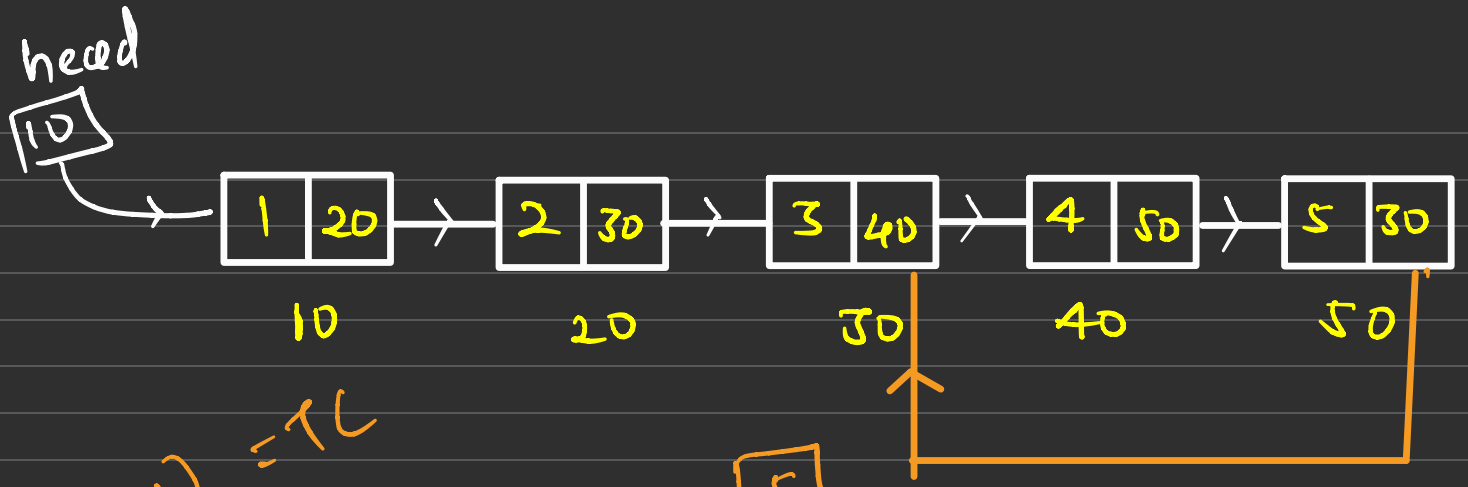
## Detect Loop in linked list



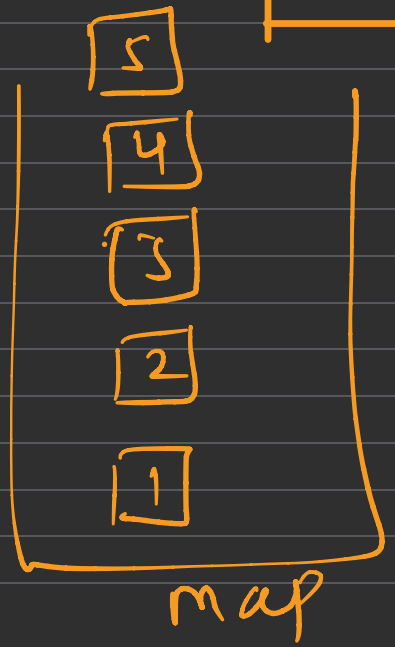


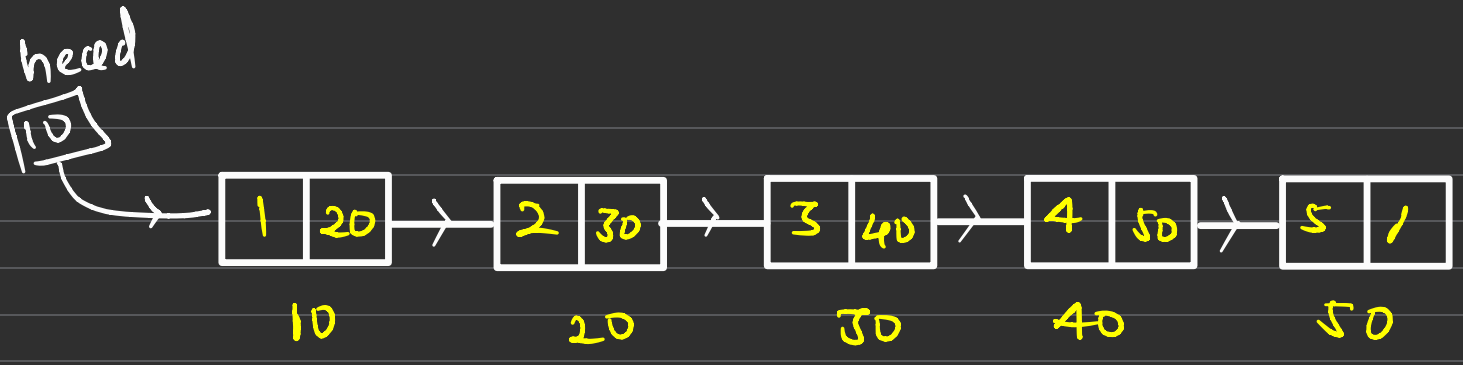


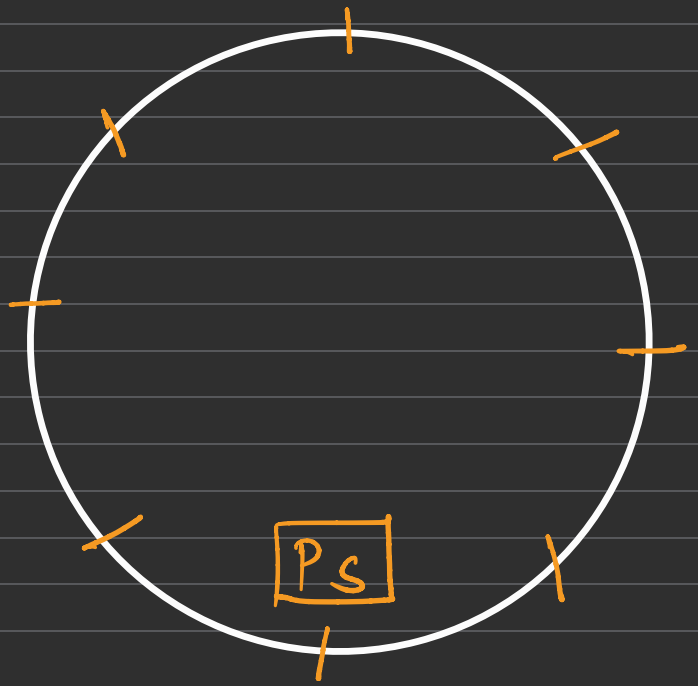

O(N) :TC

O(N) :SC



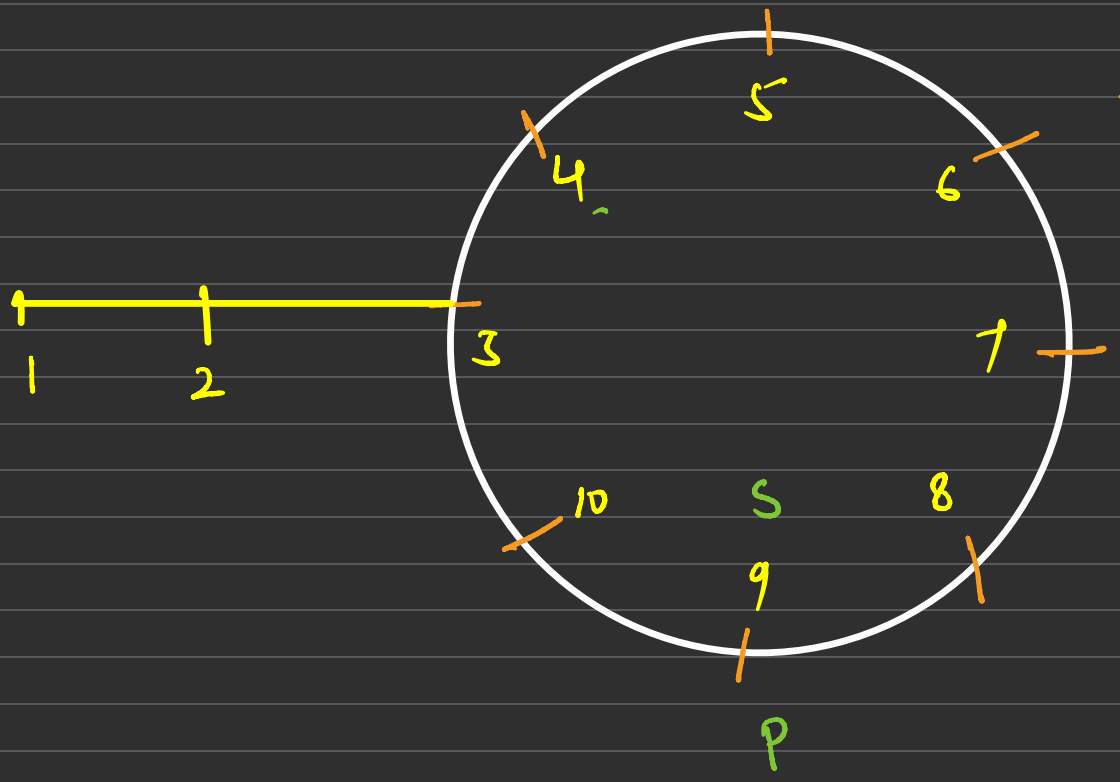

return true




P ⇒ 2

S ⇒ 1




P ⇒ 2

S ⇒ 1

P → S

d = 6

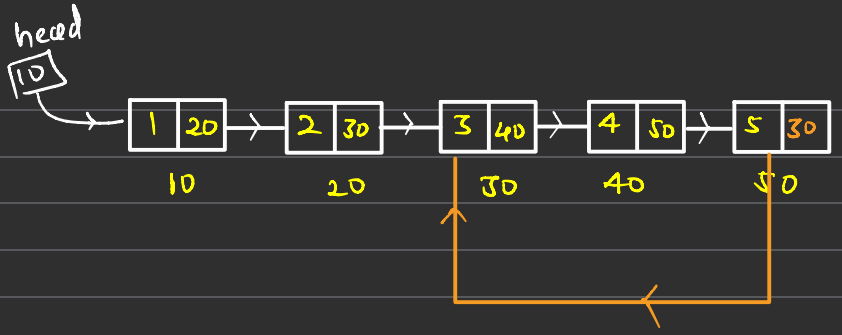
d = 5

d = 4

d = 3

d = 2

d = 0 d = 1




f

fast = head , slow = head

while ( fast != null && fast.next != null )
{
fast = fast.next.next

slow = slow.next

if ( fast == slow)
return

}
return false

## Delete without head pointer

Difficulty: **Easy** Accuracy: **78.57%** Submissions: **214K+** Points: **2**

You are given a node **del_node** of a Singly Linked List where you have to **delete** a **value** of the given node from the linked list but you are not given the **head** of the list.

By deleting the node value, we do not mean removing it from memory. We mean:

- The value of the given node should not exist in the linked list.
- The number of nodes in the linked list should decrease by one.
- All the values before & after the **del_node** node should be in the same order.

**Note:**

1. Multiple nodes can have the same values as the del_node, but you must only remove the node whose pointer del_node is given to you.
2. It is guaranteed that there exists a node with a value equal to the del_node value and it will **not be the last node** of the linked list.