Q1 Find middle of LL

Eg- $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7$

ans = 4

Brute: Find the length & goto $n/2$

using counter

Idea: Tortoise & Rabbit Algorithm

Initially tortoise at head

rabbit at head

Now both start moving

tortoise jumps by 1

Rabbit jumps by 2

When rabbit cannot jump any further, stop.

What represents mid ? Tortoise

$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7$

T

tortoise

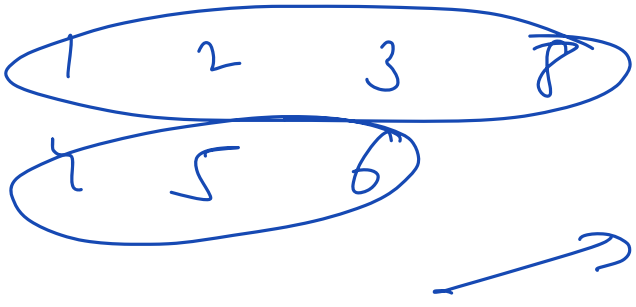
R

Code

```
Node middle ( Node head ) {
    if (head == NULL)
        return null
    slow = head          // slow= slow.next
    fast = head          // fast= fast.next.next

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

TC: O(N)

SC: O(1)



10 → 20 → 30

Q2 Merge 2 sorted LL.

(v similar to merge 2 sorted arrays)

3 → 5 → 8 → 12 P_1

2 → 4 → 6 → 7 P_2

2 → 3 → 4 → 5 → 6 → 7 → 8 → 12

```
Node merge ( Node h1 , Node h2 ) {
    Node fakehead = new Node (-1)
    temp = fakehead
    while ( h1 != null && h2 != null ) {
        if ( h1.data < h2.data ) {
            Node x = new Node (h1.data)
            temp.next = x
            temp = temp.next
        h1 = h1.next
    } else {                    same for h2 )
    }
if(h1 ==null)  temp.next = h2
else           temp.next = h1
return fakehead.next
```

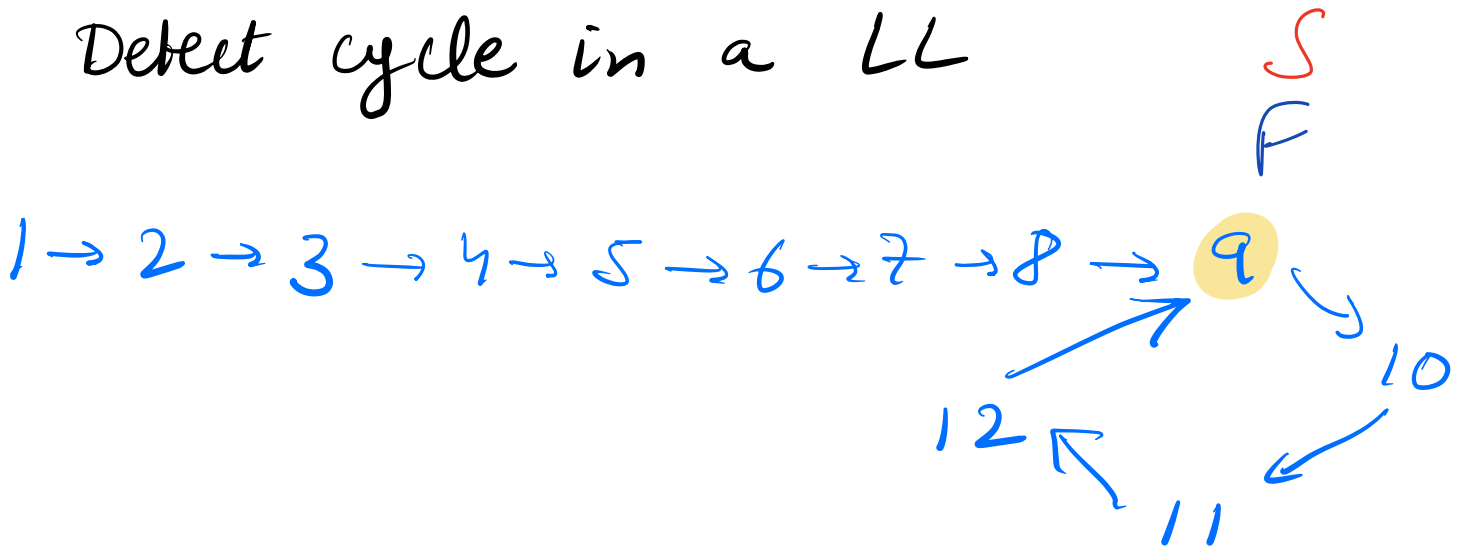
Sort the LL (mergesort)

```
Node mergesort (Node head){
    if (head==null || head.next == null)
        return head
    Node m = middle(head)
    Node h2 = m.next
    m.next = null
    head = mergesort(head)
    h2 = mergesort(h2)
    head = merge(head, h2)
    return head
}
```

5 6 7 8 ; 1 2 3 8

1 , 2 , 3 , 5 , 6 , 7 , 8 , 8

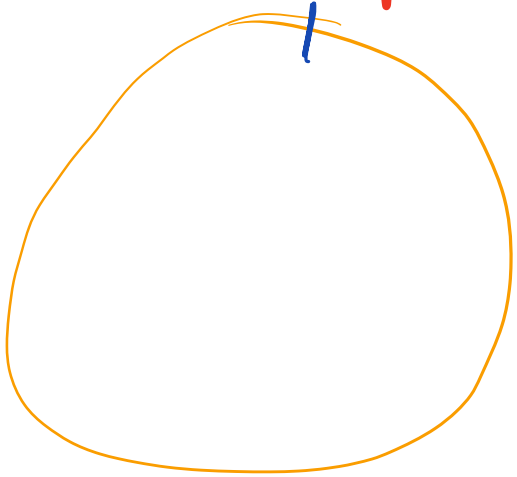
03 Detect cycle in a LL



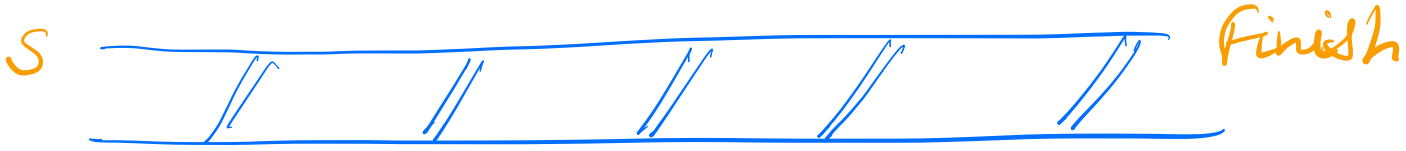
Brute: Store nodes in a hashset while iterating. If current node present in hashset return true

Idea Tortoise & Rabbit

Analogy: Imagine a running track, you & Usain Bolt start a race. Will he catch up to you at some point ?



If it is a straight track



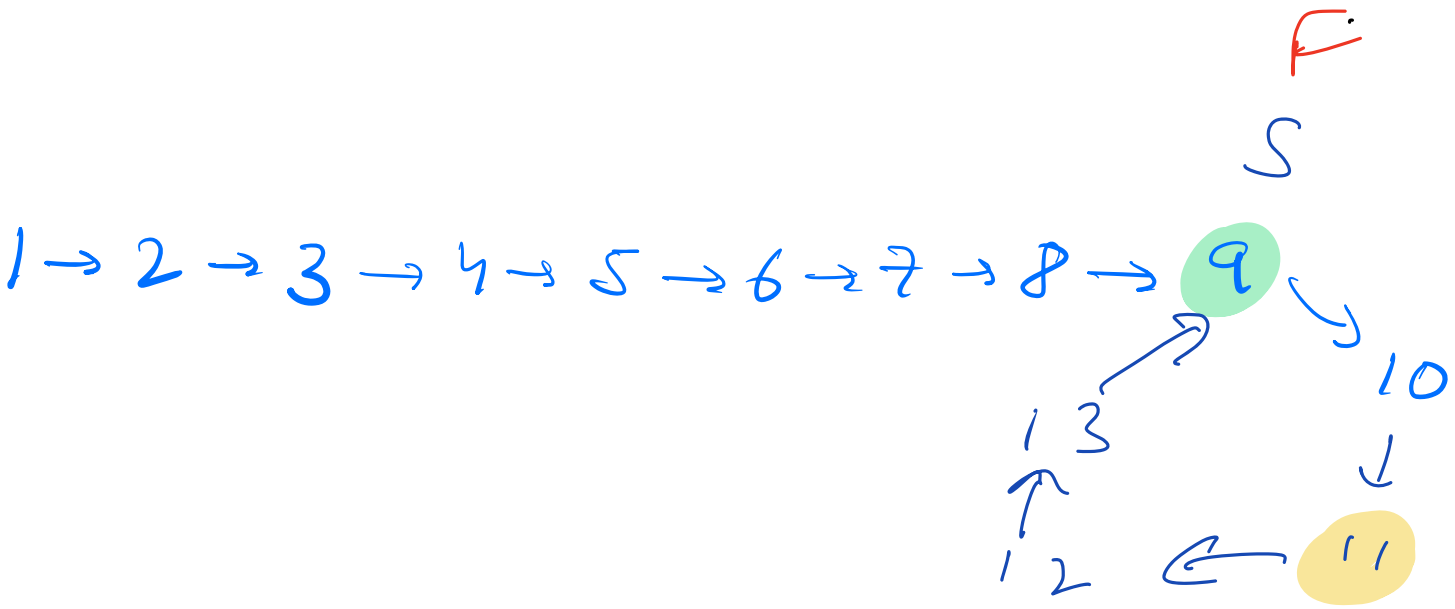
Usain Bolt just wins & goes home.

So the fast pointer will catch up to the slow pointer in a cycle.

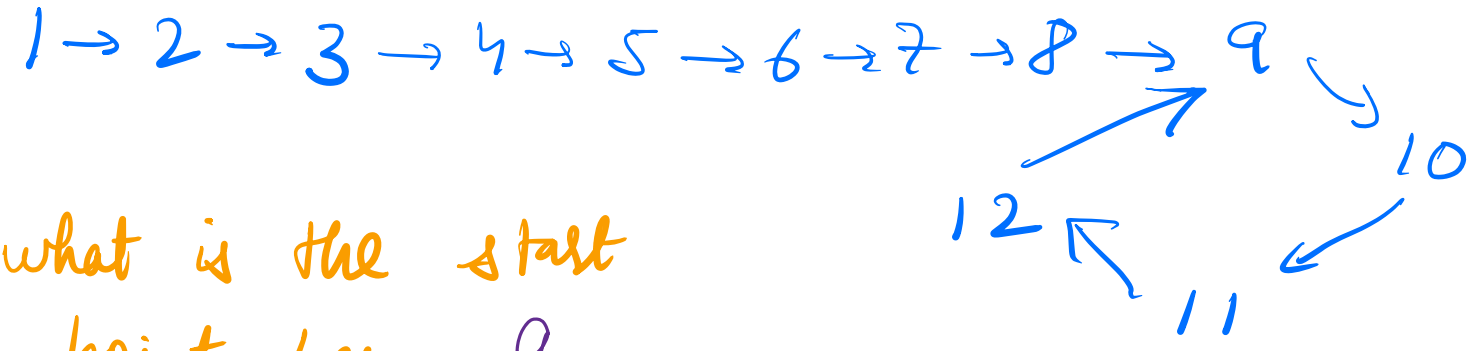
Code

```
slow = head        fast = head
while ( fast.next != NULL &&
        fast.next.next != NULL ){
    slow = slow.next
    fast = fast.next.next

    if ( slow == fast )
        return true
}

return false
```

Dry run



Q4 Find the start point of the cycle



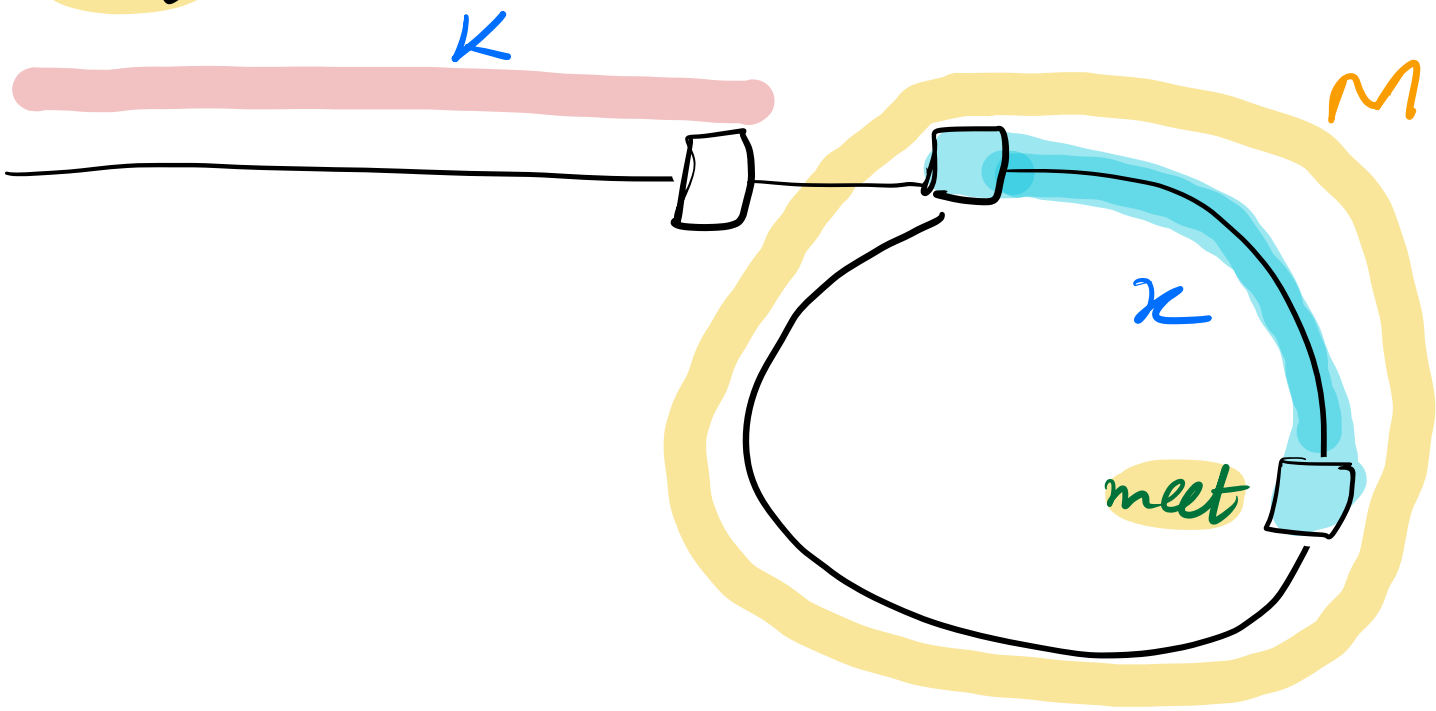
what is the start point here 9

Algorithm: First find the meeting point

Now slow = head fast = meet pt

```
while (slow != fast) {
    slow = slow.next
    fast = fast.next
}
return slow
```

Proof



Distance by slow $D_s = K + pM + x$

Distance by fast $D_f = K + qM + x$

Now $D_f = 2D_s$

$$K + qM + x = 2(K + pM + x)$$

$$K + qM + x = 2K + 2pM + 2x$$

$$M(2p - q) - x = K$$

$M(2p-q) = K+x$

$\Rightarrow$ $K+x$ is div by M

Now when slow starts from head & fast from meet,

$D_s = K$ $\qquad$ $D_f = M-x+\lambda M$

λ is random

to prove $D_s = D_f$

$K = M-x+\lambda M$

$K+x = (\lambda+1)M$

$\Rightarrow$ $K+x$ is div by M

let fh.next