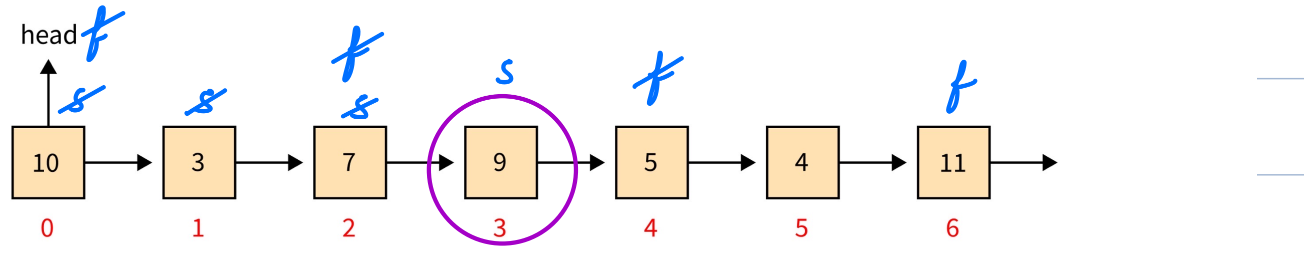
SCALER

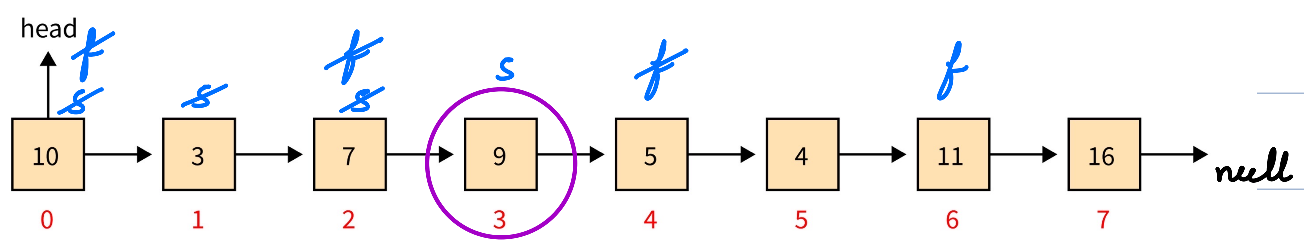
LinkedList - 2

TABLE OF CONTENTS

1. Revision
2. Middle point in Linked-list
3. Merge two sorted Linked-list
4. Merge sort in Linked-list
5. Check if there is a loop in Linked-list
6. Find the start point of the Loop

Middle of a Linked-list



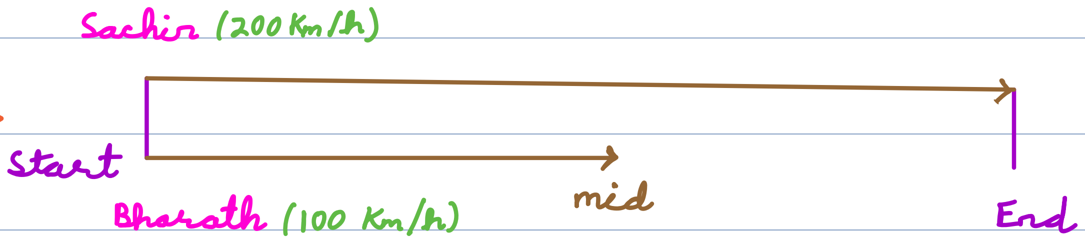


Sol 1 → 1) Find length of linked list (N). → O(N)

2) Travel half length (N/2). → O(N/2)

TC = O(N)   SC = O(1)

Sol 2 →

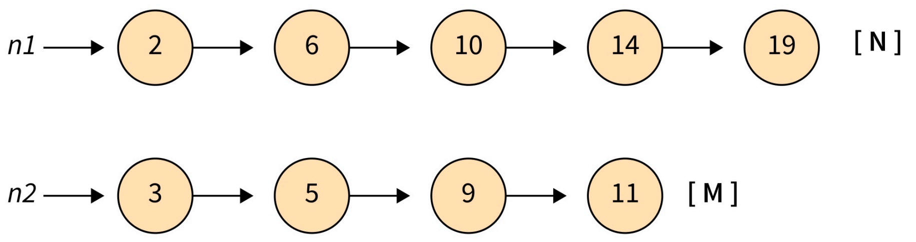


```
if (Head == null) return null
s = f = Head
while (f.next != null && f.next.next != null) {
        s = s.next
        f = f.next.next
} return s
```

TC = O(N/2) = O(N)   SC = O(1)

SCALER

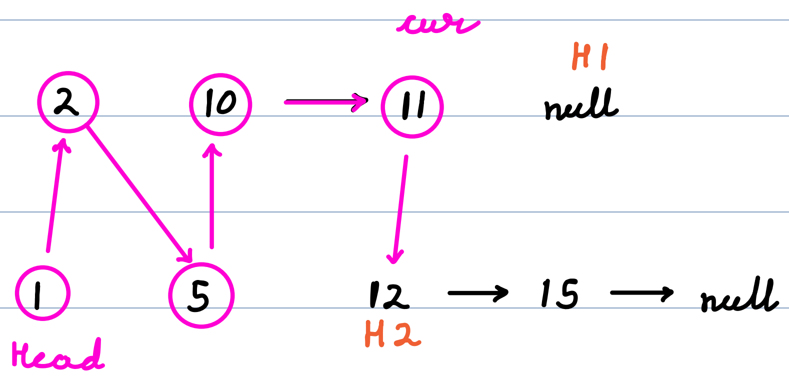
Merge Two Sorted Linked-list



Head 2 ⟶ 3 ⟶ 5 ⟶ 6 ⟶ 9 ⟶ 10 ⟶ 11 ⟶ 14 ⟶ 19 ⟶ null

H1 2 ⟶ 10 ⟶ 11 ⟶ null

H2 1 ⟶ 5 ⟶ 12 ⟶ 15 ⟶ null



if (H1 == null) return H2

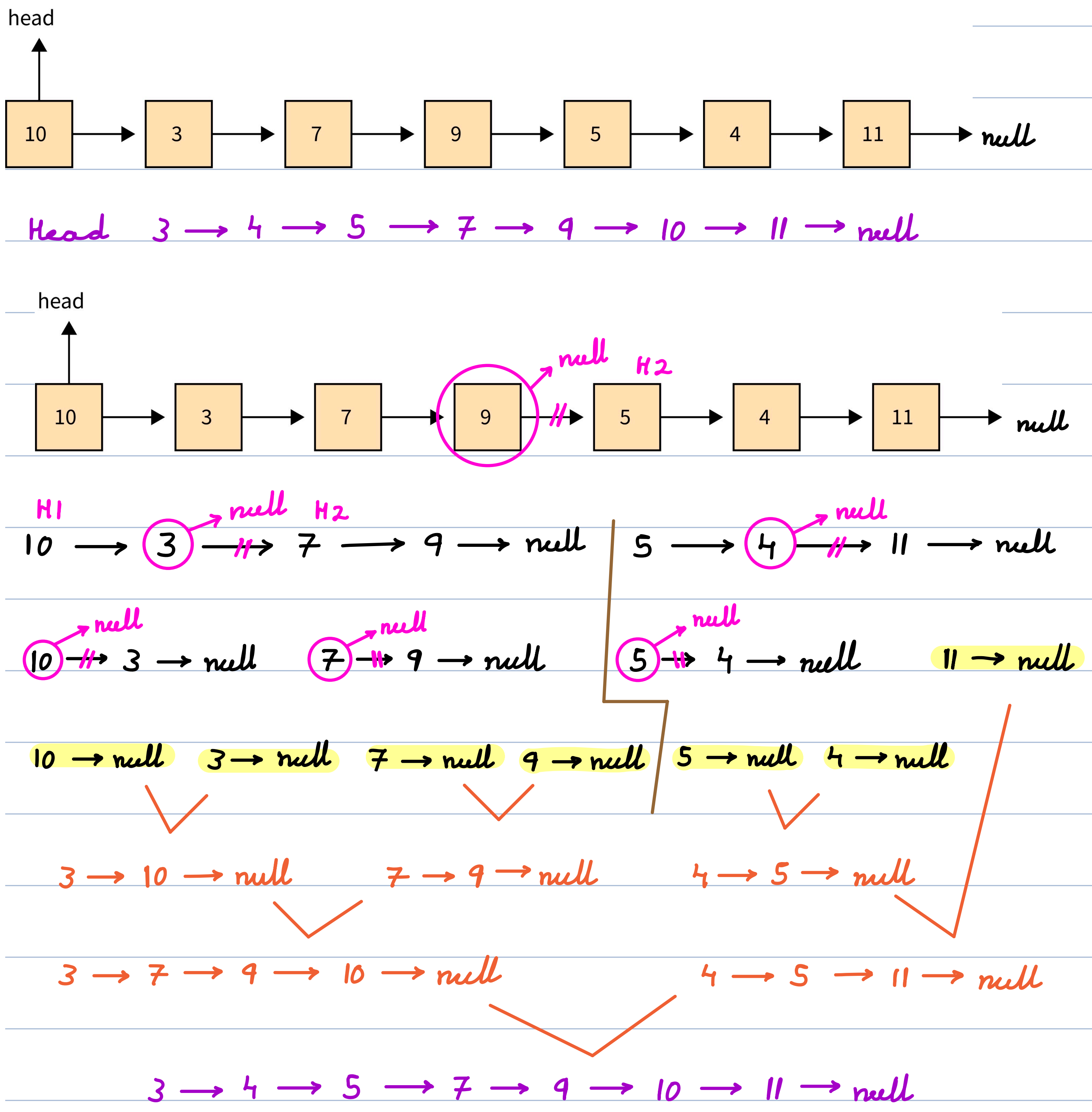
if (H2 -- = null) return H1

Head = null

```
if ( H1.data <= H2.data) {
        Head = H1
        H1 = H1.next
} else {
        Head = H2
        H2 = H2.next
}

cur = Head
while ( H1 != null && H2 != null) {
        if ( H1.data <= H2.data) {
                cur.next = H1
                H1 = H1.next
        } else {
                cur.next = H2
                H2 = H2.next
        }  cur = cur.next
}

if (H1 == null)     cur.next = H2
if (H2 == null)     cur.next = H1
return  Head
```

TC = O(N+M)   SC = O(1)

Merge Sort a Linked-list



Idea

1. Find the middle node
2. Make recursive calls to sort 1st half and 2nd half
3. Finally, merge two sorted linked-list

```
Node sort (Head) {
    if (Head == null || Head.next == null)
            return Head
    mid = findMiddle (Head)  → TC = O(N)
    H2 = mid.next
    mid.next = null
    H1 = sort (Head)
    H2 = sort (H2)
    return mergeSortedLists (H1, H2) → TC = O(N)
}
```

$TC = \underline{O(N \log(N))}$ $SC = \underline{O(\log(N))}$

Scenerio

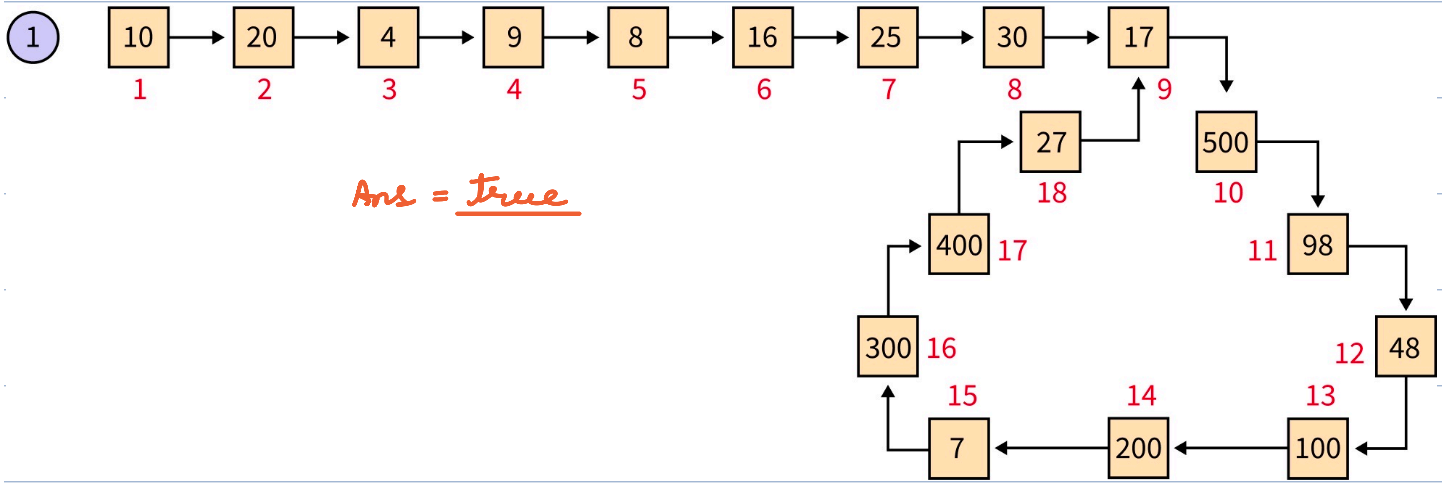
You are using **Google Maps** to help you find your way around a new place. But, you get lost and end up walking in a circle. **Google Maps** has a way to keep track of where you've been with the help of special **sensors**.

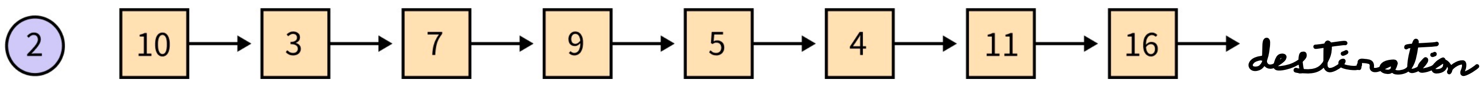
These sensors notice that you're **walking in a loop**, and now, **Google** wants to create a new feature to help you figure out exactly where you started going in circles.

Problem

You have a **linked list** that shows each **step** of your **journey**, like a chain of events. Some of these steps have accidentally led you back to a place you've already been, making you **walk in a loop**. The goal is to find out the exact point where you first started walking in this loop.

Check if there is a loop





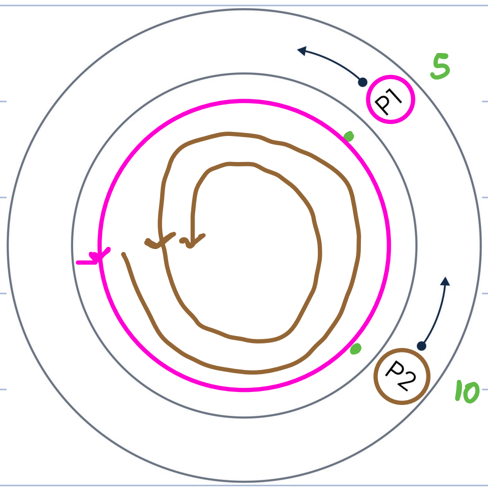
Ans = false

Sol 1 → Use hashset to check already visited node. ✓

SC = O(1)

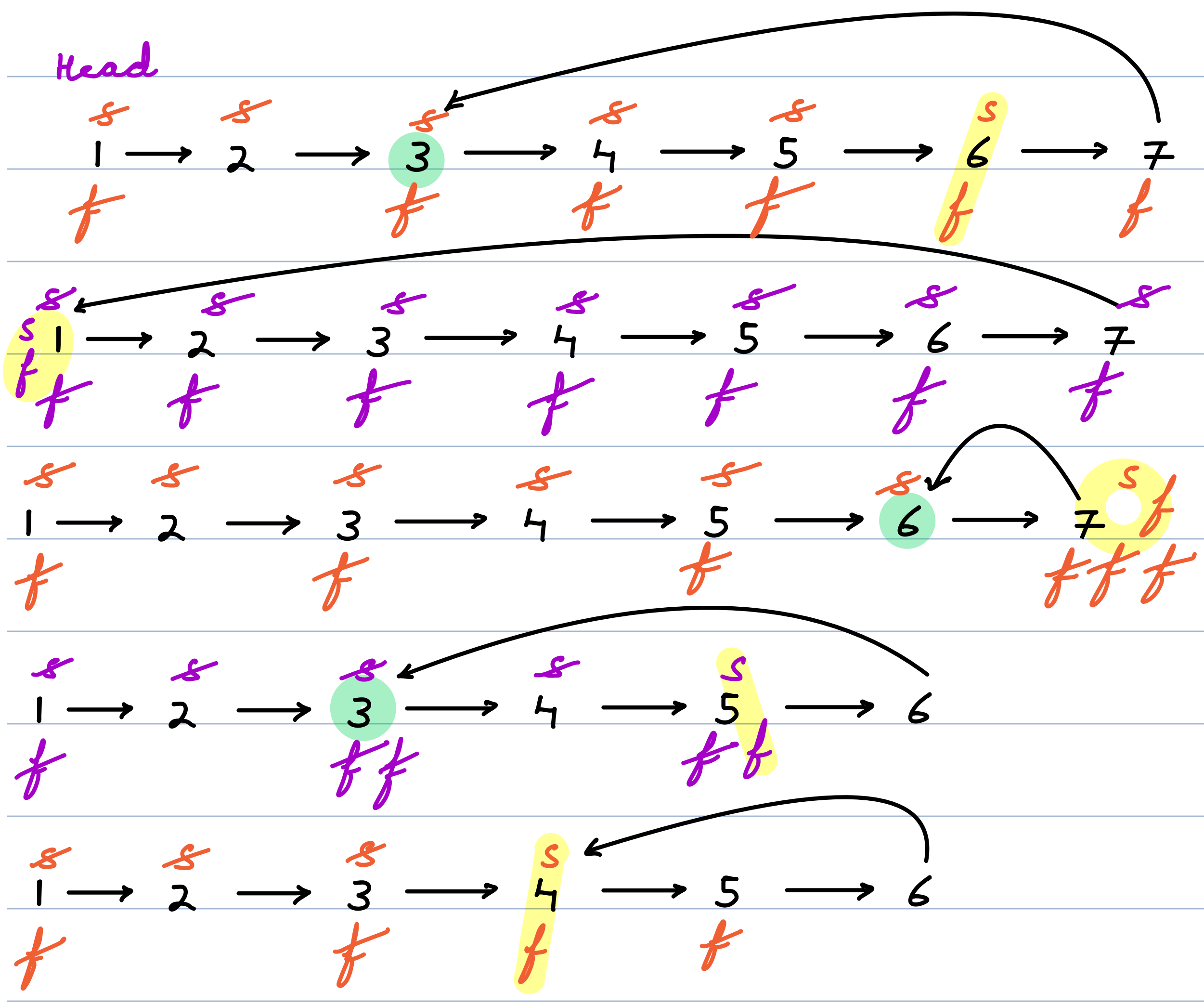
O(1)

If two people are running with different speeds on a circular track, they will 100% meet at some point.

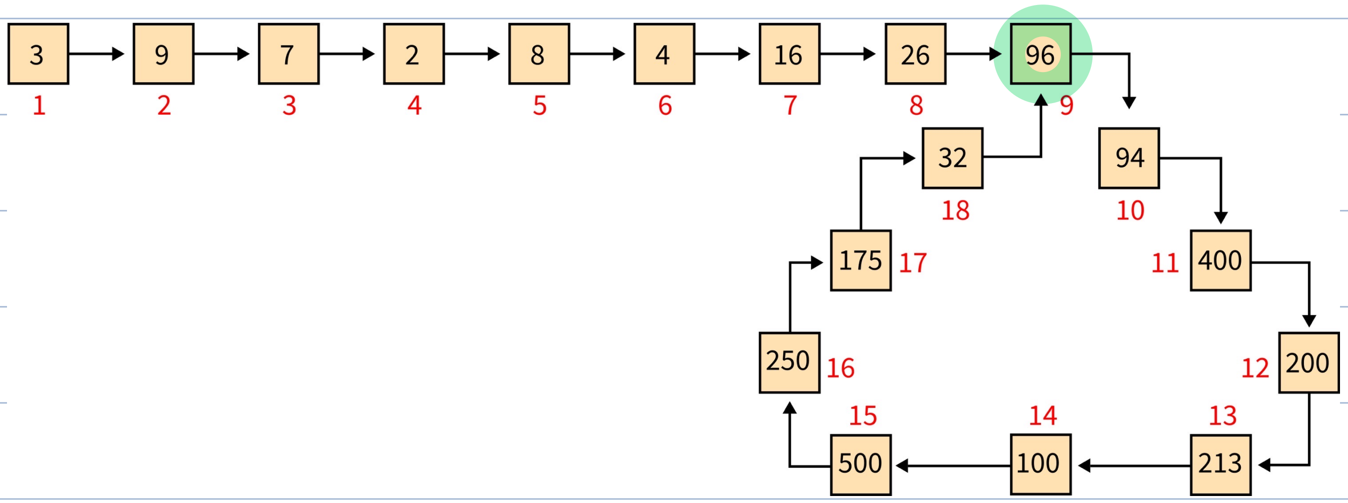


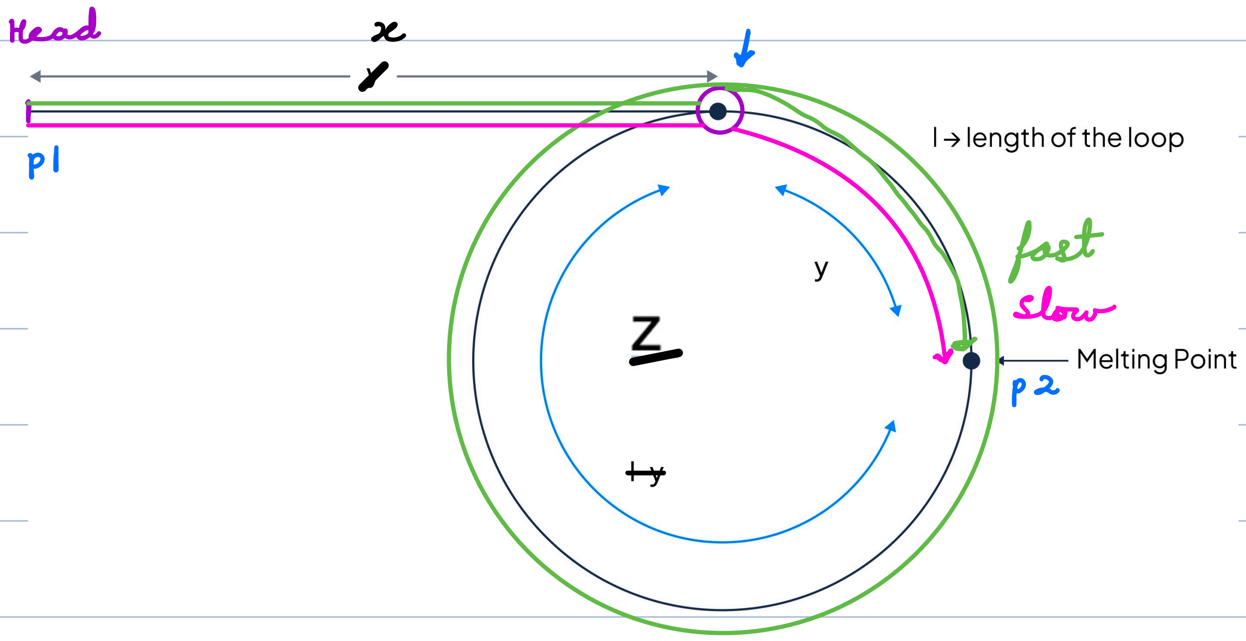
```
if (Head == null)    return false
s = f = Head
while ( f != null && f.next != null) {
        s = s.next
        f = f.next.next
        if (s == f)    return true
}

return false
```

TC = O(N)   SC = O(1)



Find the start point of the loop





<u>Distance</u>

slow = x + y

fast = x + y + z + y

Fast is all double speed ⇒

x + y + z + y = 2 * (x + y)

~~x~~ + ~~y~~ + z + ~~y~~ = ~~x~~ + ~~y~~ + x + ~~y~~

⇒ $\boxed{z = x}$

x

```
if (Head == null)    return false
s = f = Head
while ( f != null && f. next != null) {
    s = s. next
    f = f. next. next
    if (s == f)    break
}

p1 = Head
p2 = s
while (p1 != p2) {
    p1 = p1. next
    p2 = p2. next
}
return p1
```

TC = O(N)   SC = O(1)