Q→ Find the middle element of the linked list.

$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & \textcircled{3} \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & \text{null} \end{tikzcd}$$

Head

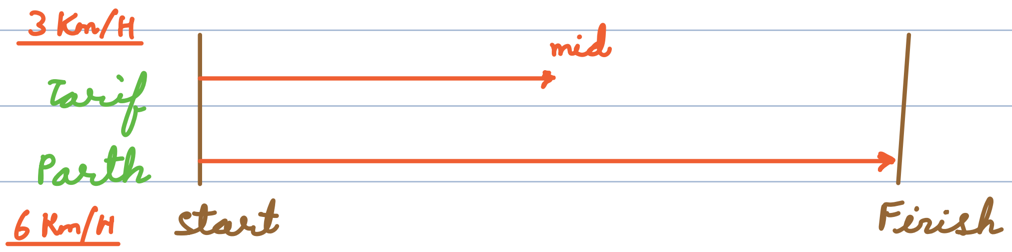
$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & \textcircled{3} \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 6 \arrow[r] & \text{null} \end{tikzcd}$$

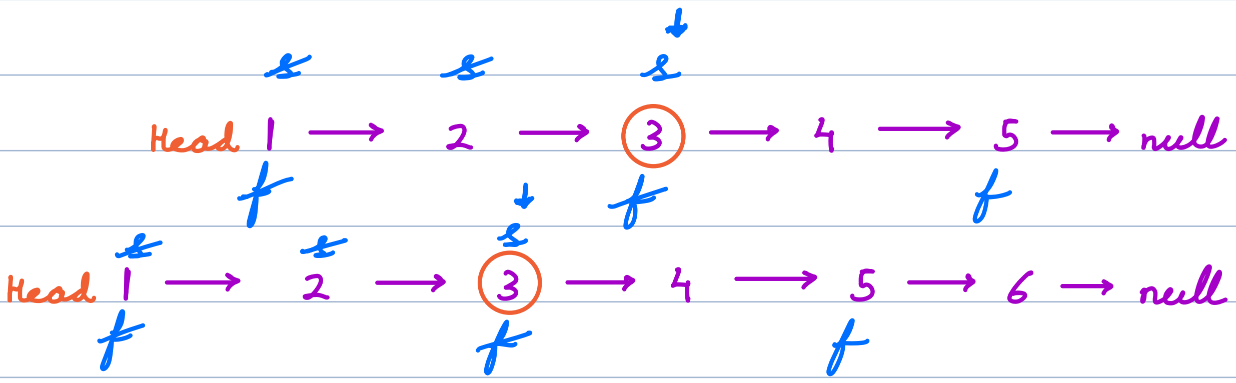
Head

<u>Sol 1</u> → 1) Find length of linked list.
2) Travel half length to find middle.

$TC = O(N + \frac{N}{2}) = \underline{O(N)}$ $SC = \underline{O(1)}$

<u>Solve in 1 traversal</u> (slow & fast pointer)





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

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

---

Q → Given 2 sorted linked list.
Merge them into a single sorted list.

H1 ... H2

1 → 2 → 8 → 10 → null | 3 → 5 → 9 → 11 → null

~~x~~ ~~x~~ ~~x~~ ~~x~~ x | ~~y~~ ~~y~~ ~~y~~ y

Head

1 → 2 → 3 → 5 → 8 → 9 → 10 → 11 → null

H1 2 → 10 → 11 → null

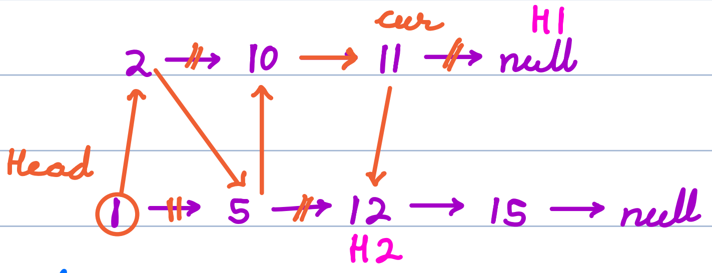
H2 1 → 5 → 12 → 15 → null

Head 1 → 2 → 5 → 10 → 11 → 12 → 15 → null

```
// H1 & H2 → i/p
if (H1 == null) return H2
if (H2 == null) return H1

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
```

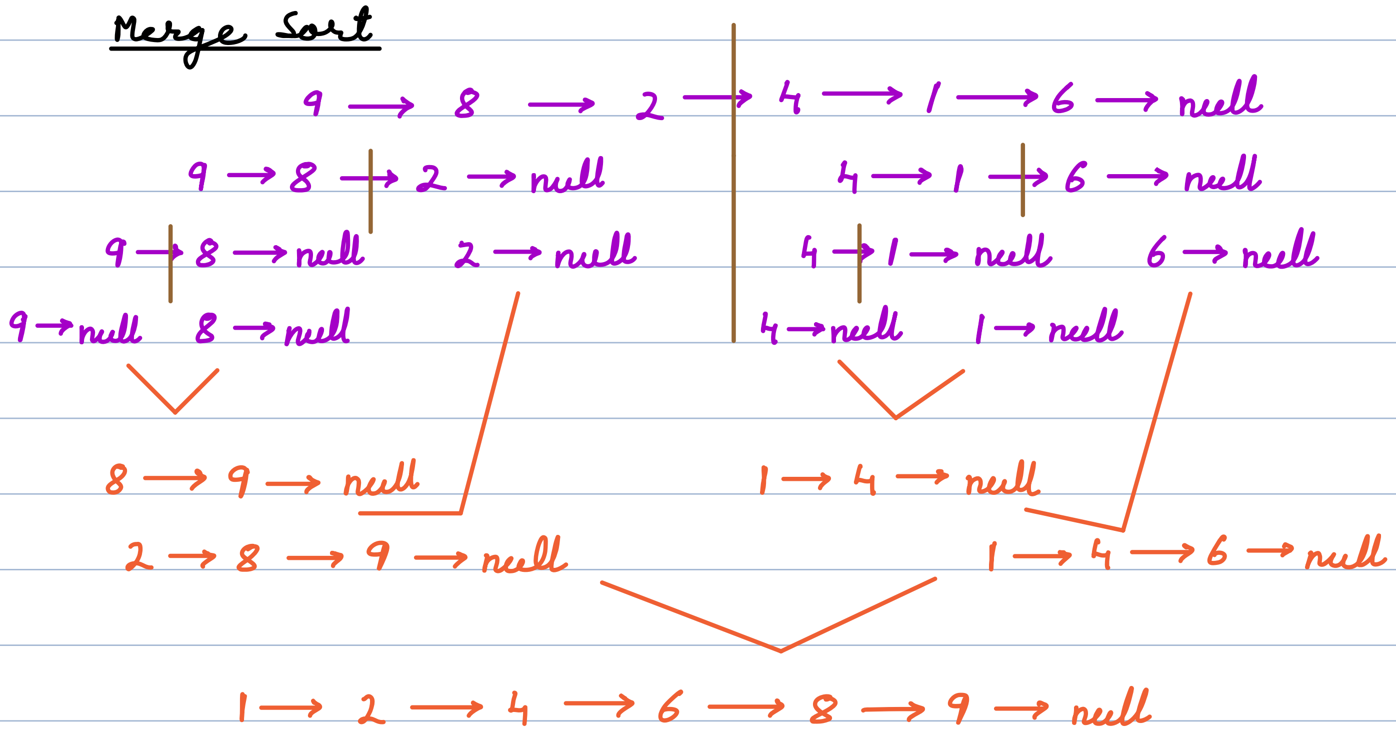


```
    } else {
        cur.next = H2
        H2 = H2.next
    }
    cur = cur.next
}
if (H1 != null)   cur.next = H1
if (H2 != null)   cur.next = H2
return Head
```

$TC = \underline{O(N+M)}$   $SC = \underline{O(1)}$

---

**Merge Sort**



```
Node sort(Head) {
    if (Head == null || Head.next == null) {
        return Head
    }
    mid = getMiddle(Head) → TC = O(N)   SC = O(1)
    H1 = Head        H2 = mid.next
    mid.next = null
    sort (H1)          sort (H2)
    Head = merge (H1, H2)  → TC = O(N)     SC = O(1)
     return Head
}
```

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

---

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 6 \arrow[lllll, bend right] \\
\text{Head} & & & & &
\end{tikzcd}$$

<u>Circular Linked List</u>

Q → check if the given linked list has a cycle.

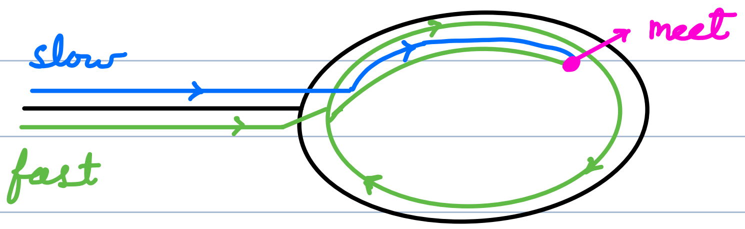
$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 6 \arrow[r] & \text{null} \\
\text{Head} & & & & & & \text{Ans} = \underline{\text{false}}
\end{tikzcd}$$

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 6 \arrow[lll, bend left] & \text{Ans} = \underline{\text{true}} \\
\text{Head} & & & & & &
\end{tikzcd}$$

<u>Sol 1</u> → Travel the list, if null found ⇒ ans = false
else ⇒ ans = true.
no stopping point ✗

<u>Sol 2</u> → Travel & keep track of visited nodes,
if null found → ans = false ✓
else if a vst node repeats → ans = true
TC = $\underline{O(N)}$   SC = $\underline{O(N)}$ // HashSet

Sol 3 →



```
if (Head == null) return false
s = Head        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)
```

---

Q → Given a linked list with cycle,
find the start of cycle.

$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & \boxed{3} \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 6 \arrow[lll, bend left] \end{tikzcd}$$

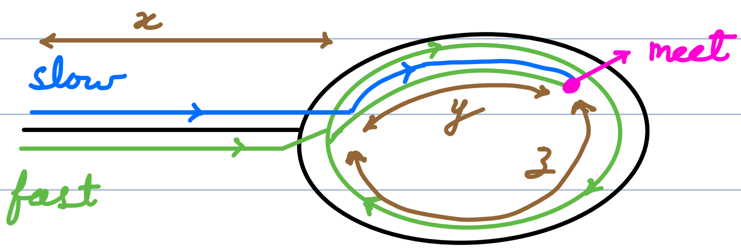
Head          Ans = 3

Sol 1 → Travel & keep track visited nodes,
if a vst node repeats → ans = that node

TC = O(N)      SC = O(N) // HashSet

→ SC = O(1)

Sol 2 →



Distance travelled by →

slow = $x + y$

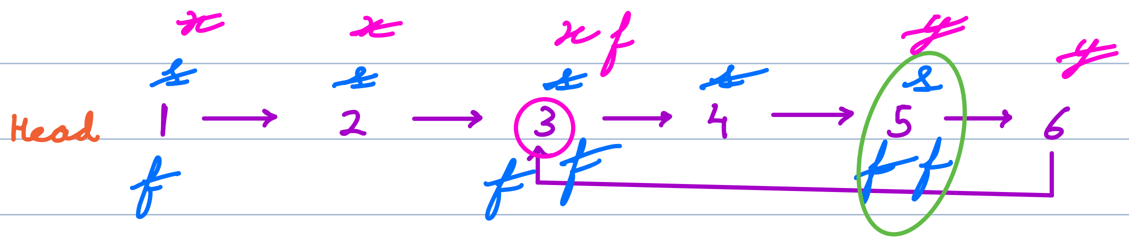
fast = $x + y + z + y$

∵ fast runs at double speed wrt slow

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

$$\cancel{x} + \cancel{y} + x + \cancel{y} = \cancel{x} + \cancel{y} + z + \cancel{y}$$

$$\Rightarrow \boxed{x = z}$$



```
s = Head        f = Head
while ( true ) {
    s = s. next
    f = f. next. next
    if ( s == f ) break
}

x = Head    y = s    //or f
while (x != y) {
    x = x. next
    y = y. next
}
return x  //or y
```

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