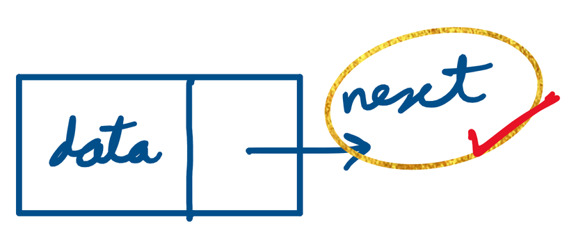
Linked List

Monday, 13 September 2021 9:04 PM

Limitations on arrays →
1) Fixed size
2) Insertion
3) Deletion

} Dynamic Arrays ✓

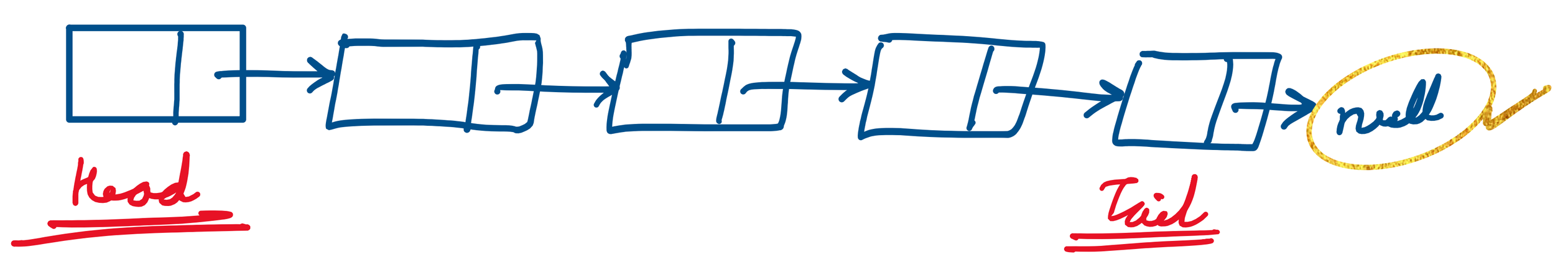
Linked List




```
struct node {
    int data;
    node * next;
}
```

```
class node {
    int data;
    node next;
}
```

Non-continuous Memory Allocation



Operations in LL

Never modify head pointer unless important.

1) Access k^{th} element → Array → A[k-1] TC = O(1)

```
node x = head;
for (i = 1; i < k; i++) {
  ✓→ if (x == null) return null;
     x = x → next;
}
return x;
```

TC = O(K) ≈ O(N) ✓

2) Search a value v → Array
- → Linear Search TC = O(N)
- → Binary Search (only Sorted Arrays) TC = O(log n)

```
node x = head;
while ( x != null) {
    if (x → data == v)  return x;
    x = x → next;
}
return null;
```

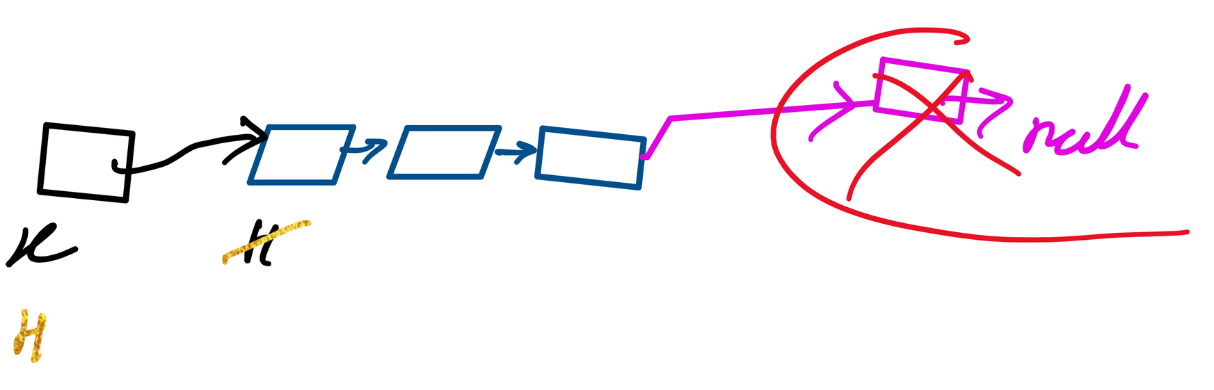
TC = O(N)

3) Insertion a value v → Array → Make a new array of size N+1 & copy all N contents. TC = O(N) ✓

Insert head

```
node x = newNode(v);
x → next = head;
H = x;
return H;
```

TC = O(1) ✓



Insert end

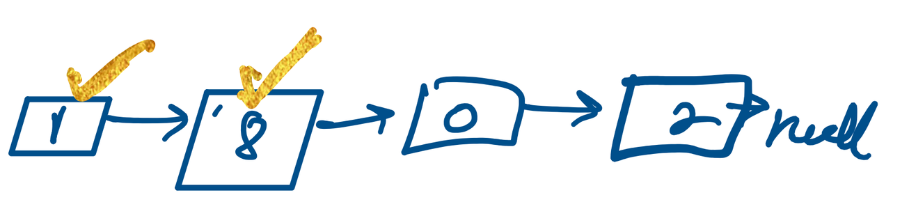
```
node x = head;   //head != null
while (x → next != null) {
      x = x → next;
}
x → next = newNode(v);
```

TC = O(N) ✓

4) Deletion of value (v) → Array → O(N) ✓

```
node delete ( H, v) {
    if ( H == null)  return null;
 →  if (H → data == v)  return head→next;
    node x = head;
    while( x → next != null) {
        if (x → next → data == v) {
            x → next = x → next → next;
            return head;
        }
        x = x → next;
    }
    return head;
```

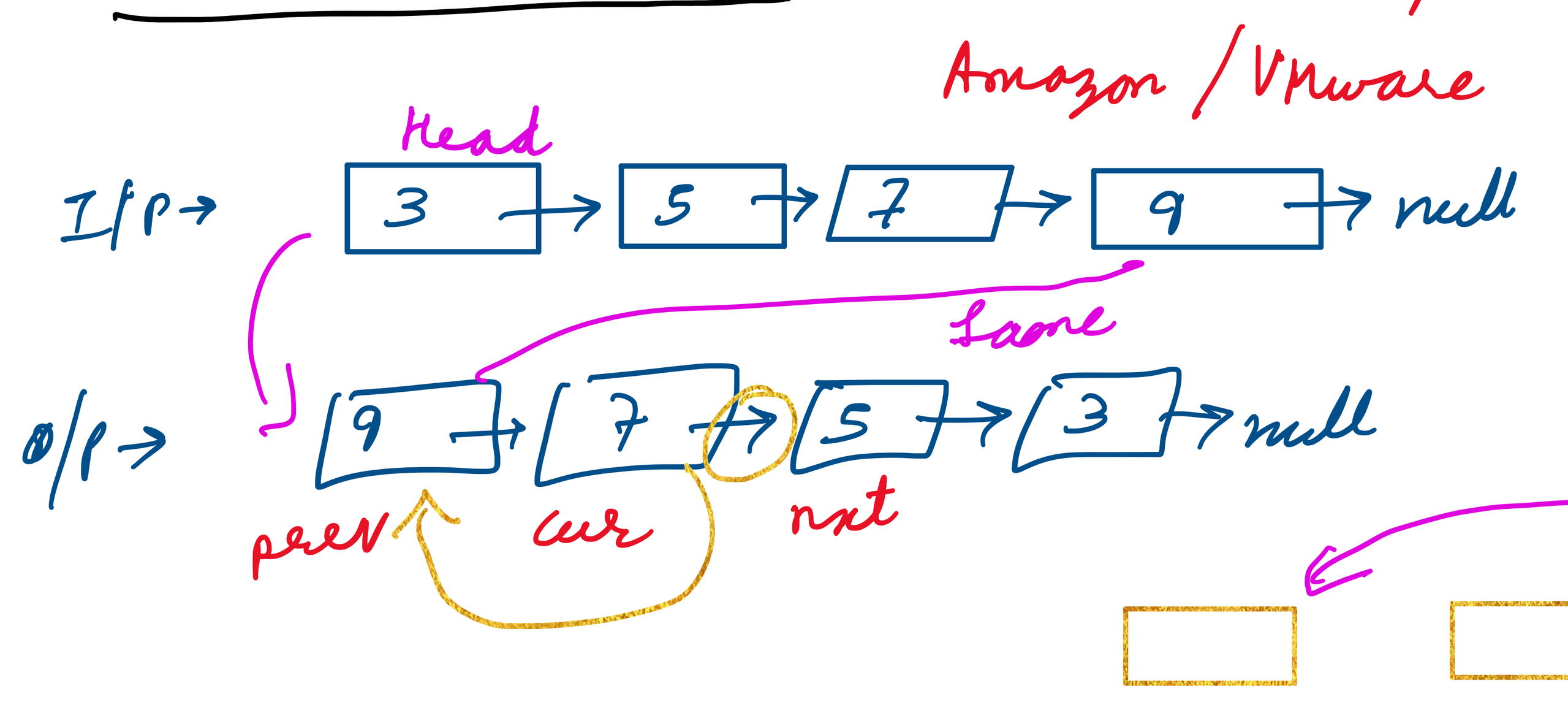
(H → LL)



TC = O(N)

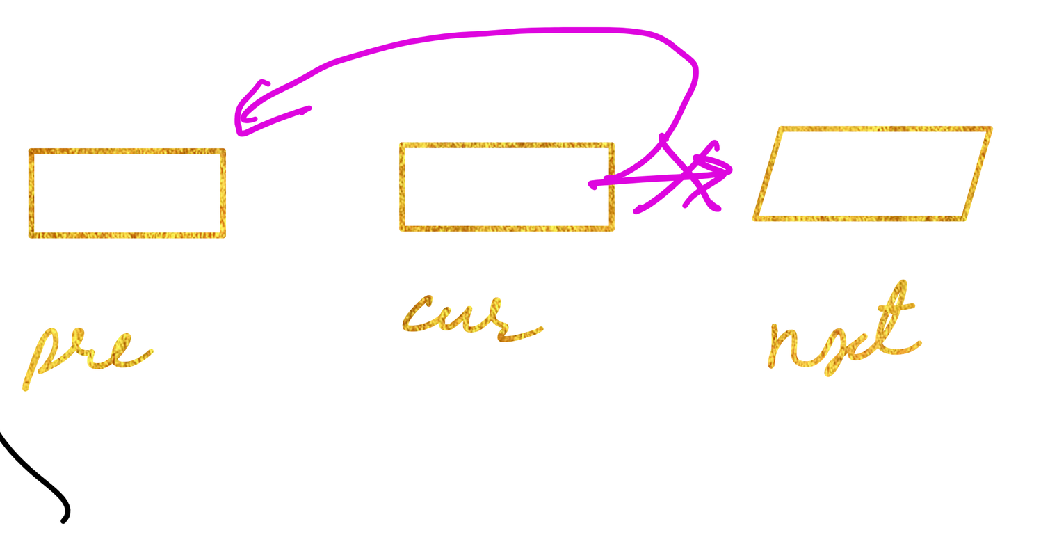
Q→ Reverse Linked List

Qualcomm / Expedia / Wipro / HCL / SAP labs
Amazon / VMware



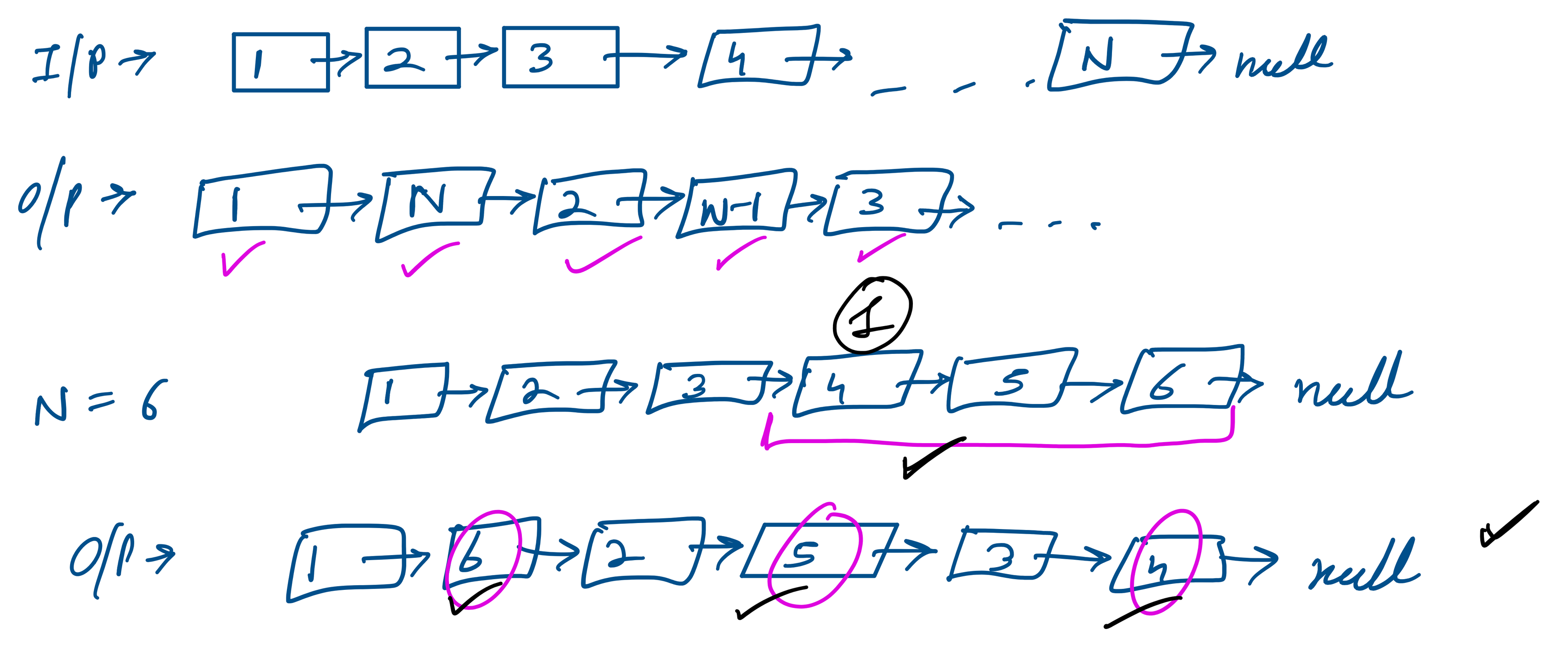

```
node pre = null, nxt = null;
node cur = head;
while (cur != null) {
    nxt = cur → next;
    cur → next = pre;
    pre = cur;
    cur = nxt;
}
return pre;
```

TC = O(N)

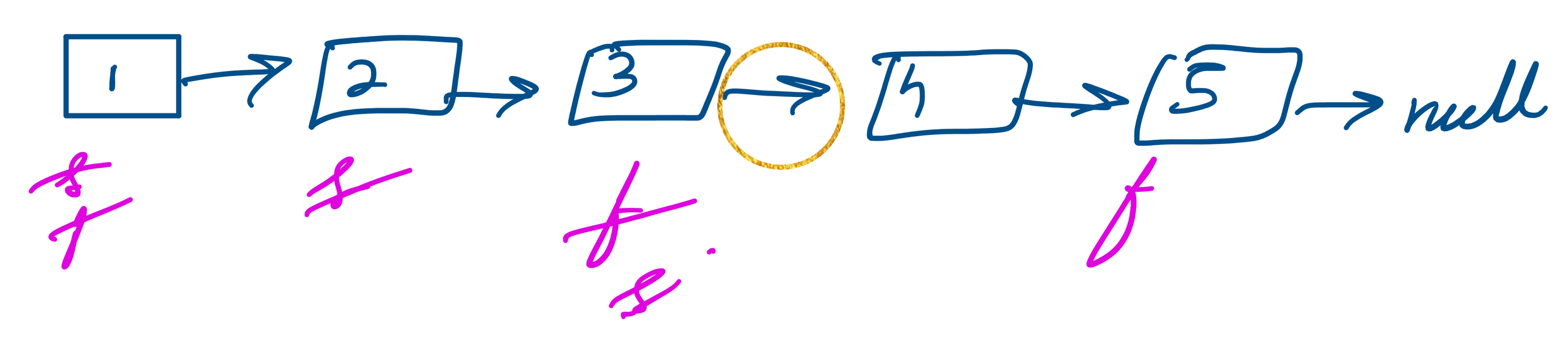


pre → 9
cur → null
nxt → null

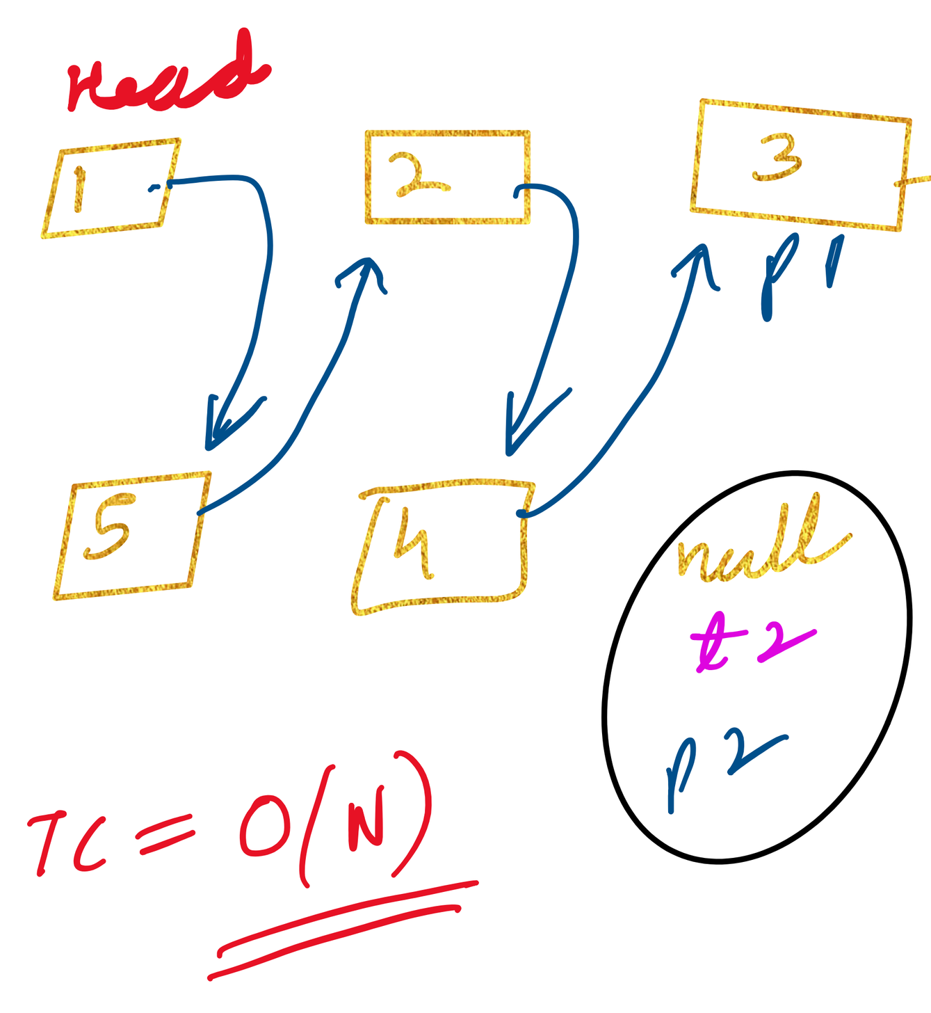
Q→ Reorder List



Middle element → 1) calculate N & middle = $\frac{N}{2}$
2) Slow & Fast Pointers
↓ +1   ↓ +2



N/2
s = s → next;
f = f → next → next;
N



TC = O(N)

```
t1 = p1 → next;
t2 = p2. next;
p1 → next = p2;
p2 → next = t1;
p1 = t1;
p2 = t2;
```