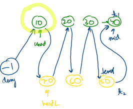
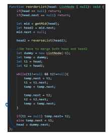
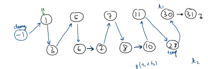


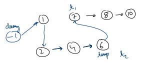


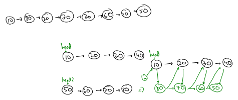
```
function reorderList(head: ListNode | null): void {
    if(head == null) return;
    if(head.next == null) return;

    let mid = getMid(head);
    let head2 = mid.next;
    mid.next = null;

    head2 = reverseList(head2);

    //We have to merge both head and head2
    let dummy = new ListNode(-1);
    let temp = dummy;
    let t1 = head;
    let t2 = head2;

    while(t1!=null && t2!=null){
        temp.next = t1;
        t1 = t1.next;
        temp = temp.next;

        temp.next = t2;
        t2 = t2.next;
        temp = temp.next;
    }

    if(t1 == null) temp.next= t2;
    else temp.next = t1;
    head = dummy.next;
};
```



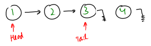
dummy.next

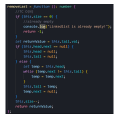
```
if (t1 < t2)
    temp.next = t1
    t1 = t1.next
    temp = temp.next
```



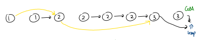


1) Mid
2) Break
3) l2 reverse
4) Merge





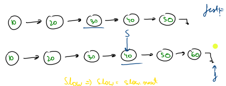
```
removeLast = function (): number {
    //TC O(N)
    if (this.size == 0) {
        //already empty
        console.log("Linkedlist is already empty!");
        return -1;
    }
    let returnValue = this.tail.val;
    if (this.head.next == null) {
        this.head = null;
        this.tail = null;
    } else {
        let temp = this.head;
        while (temp.next != this.tail) {
            temp = temp.next;
        }
        this.tail = temp;
        temp.next = null;
    }
    this.size--;
    return returnValue;
};
```



```
while( curr != null) {
    let temp = curr. next;
    while( curr.val == temp.val ){
        temp = temp.next
    }
    curr.next = temp

    curr = curr.next
}
```

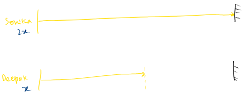
fast.next = null

fast != null

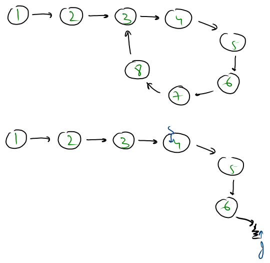


Slow => Slow = slow.next

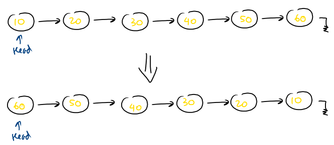
fast => fast = fast.next.next

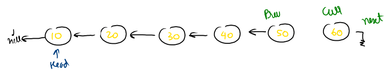


Detect cycle in linked list



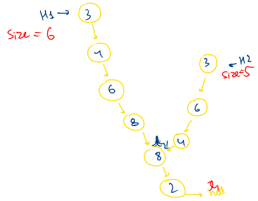
Reverse LL





```
curr.next = prev
Prev = curr
curr = next
next = next.next
```

Intersection Point of LLs



1) get size for both
2) Bigger LL headstart
3) Move both point together
4) Same node = ans