

## Introduction to Linked Lists : →

It is a data structure containing entities called "Nodes". The nodes can be of many types :

- \* 

Data	Next
------	------

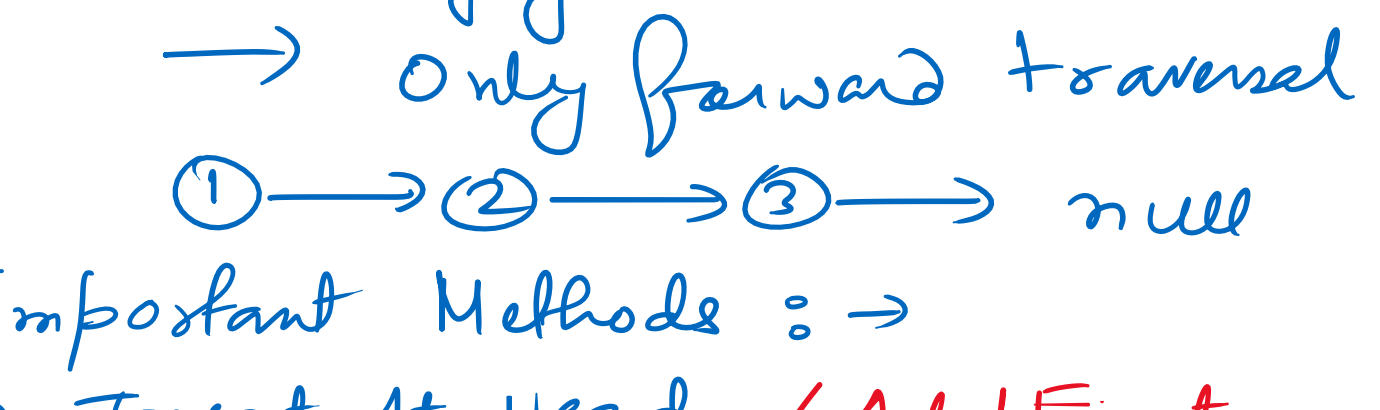
 → only linked to next node.
- \* 

Prev	Data	Next
------	------	------

 → linked to next & prev nodes.

Based on this, there are three types.

- \* Singly linked list
- \* Doubly linked list
- \* Circular linked list
- \* XOR linked list

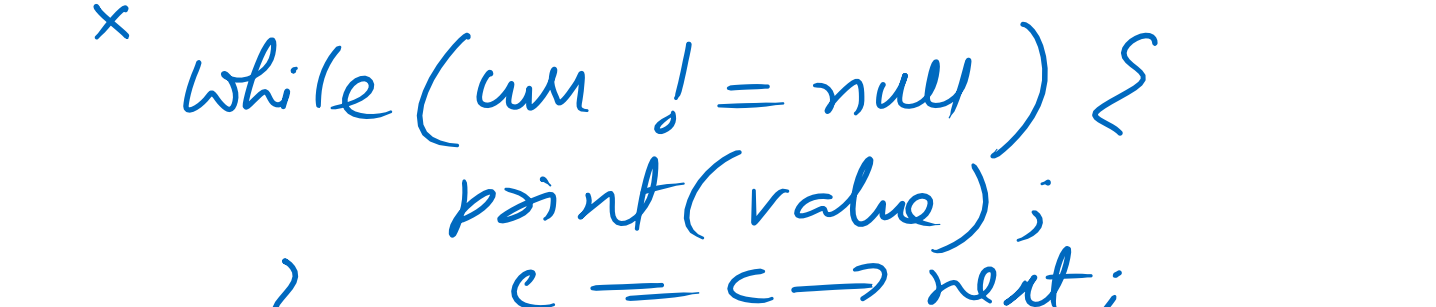


h → t → Singly Linked List  
→ Only forward traversal

① → ② → ③ → null

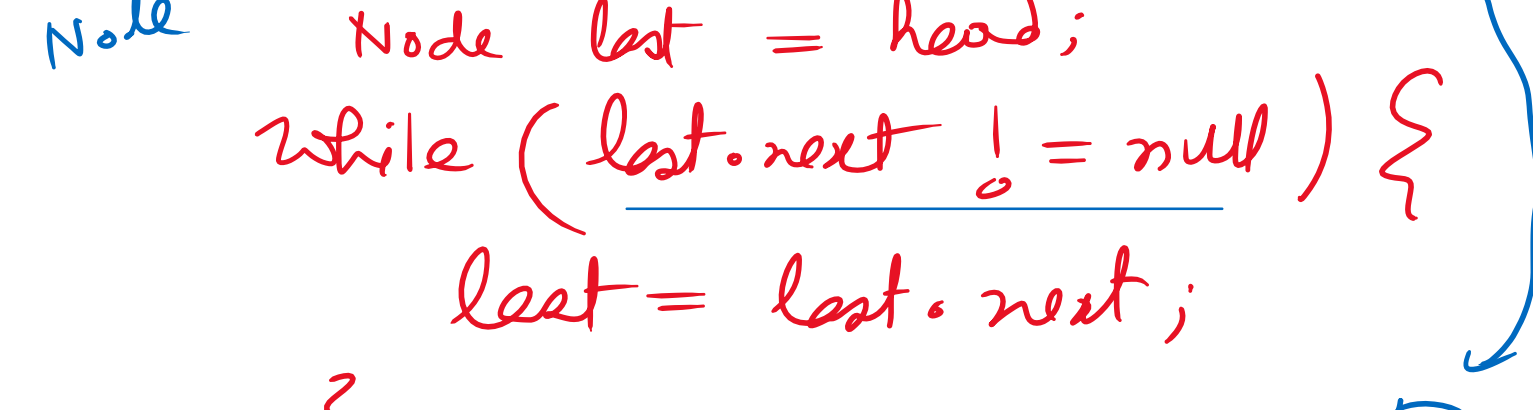
## Important Methods : →

- I (Insert)
  - ① Insert at Head / Add First
  - ② Insert at Tail / Add Last
  - ③ Insert After Specific
- D (Delete)
  - ④ Delete First / head
  - ⑤ Delete Last / Tail
  - ⑥ Delete Target



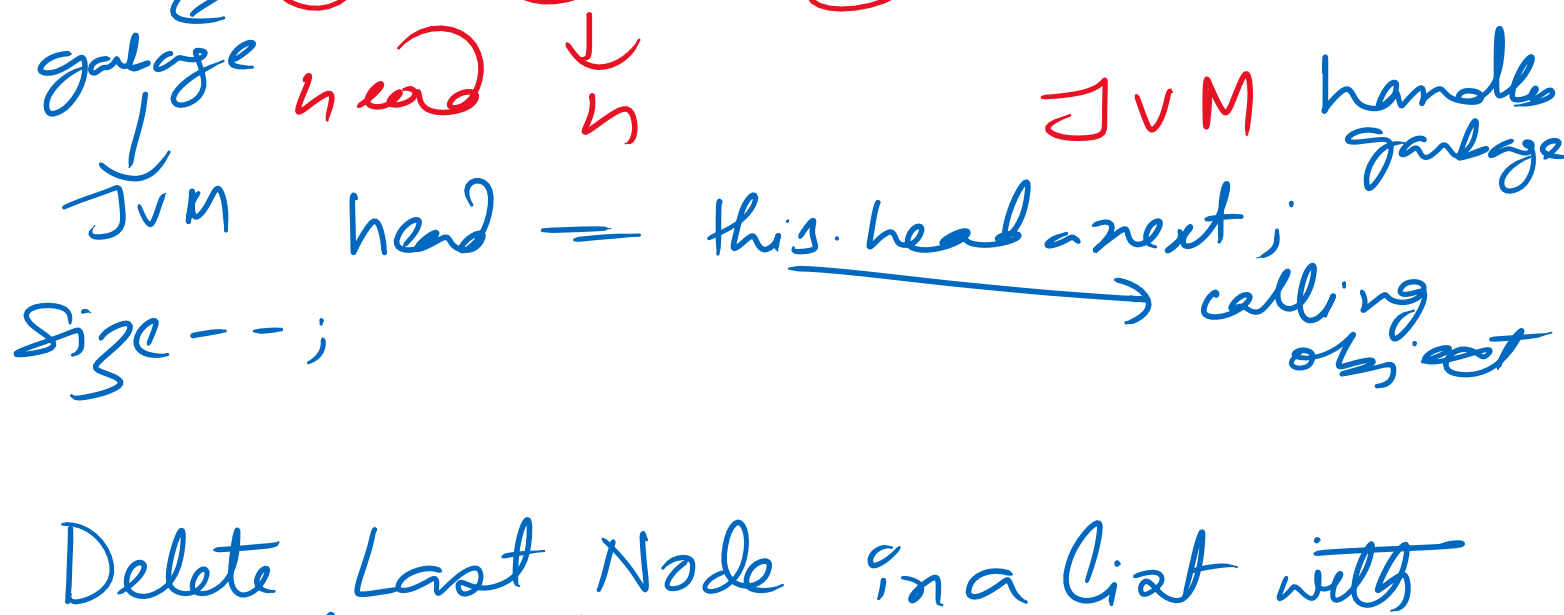
while (curr != null) {  
    print(value);  
    curr = curr → next;  
}

Add First : → new Node (X) → A → B → C → null  
nN.next = head  
h = nN



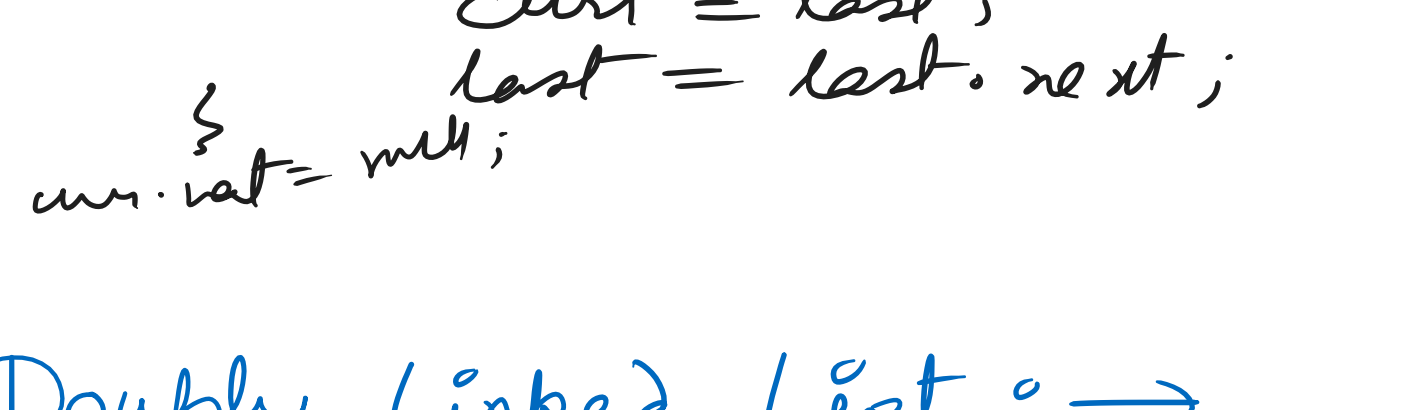
Node last = head;  
while (last.next != null) {  
    last = last.next;  
}  
last.next = new Node;

delete() free()  
head = head.next;



Size--;  
head = this.head.next; calling object

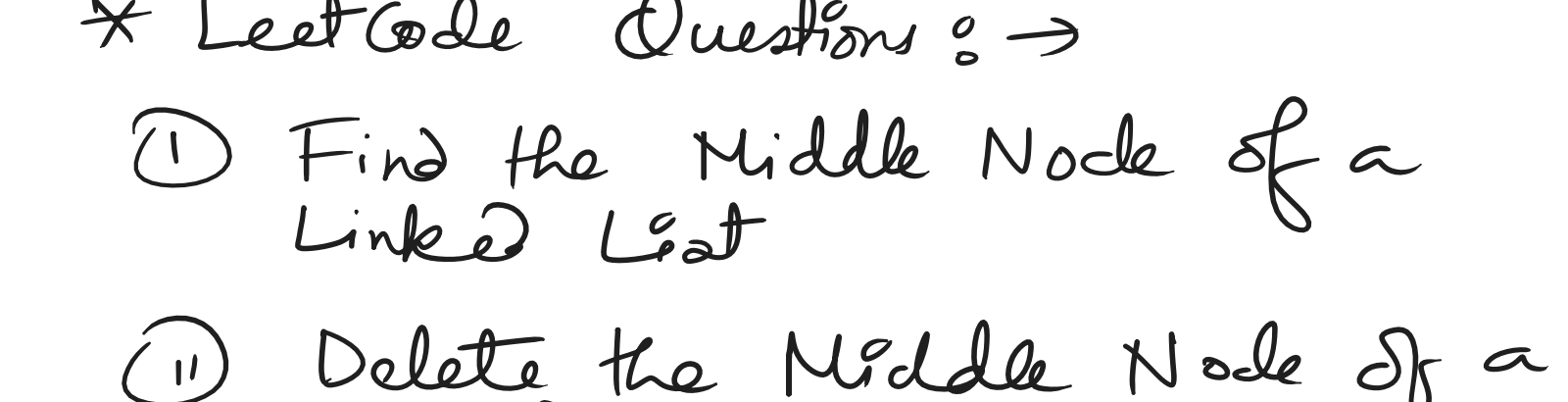
## Delete Last Node in a List with multiple nodes.



curr = head  
last = head.next  
while (last.next != null) {  
    curr = last;  
    last = last.next;  
}  
curr.next = null;

## Doubly Linked List : →

Both Forward & Backward Traversal

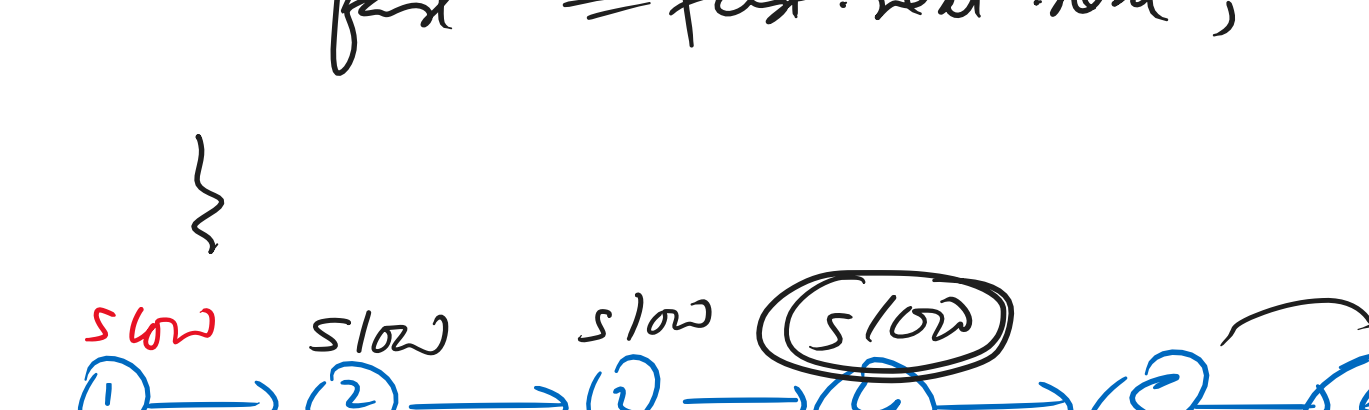


head.prev = null  
tail.next = null  
Node structure

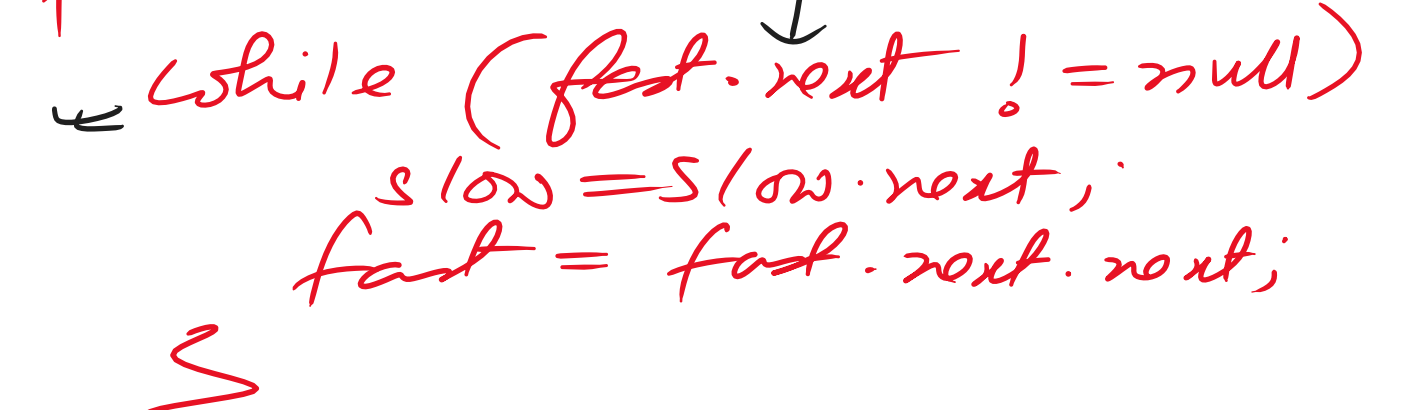
```
class Node {  
    Node next;  
    Node prev;  
    int data;  
}
```

## \* LeetCode Questions : →

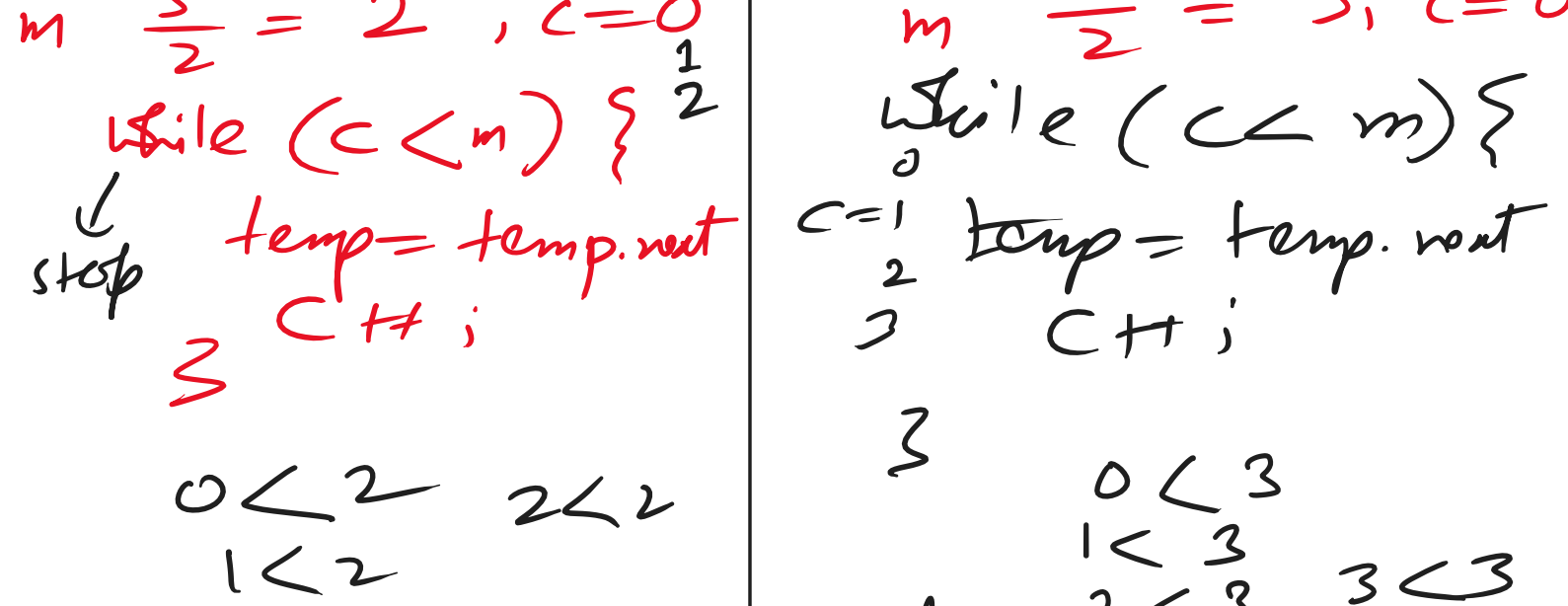
- ① Find the Middle Node of a Linked List
- ② Delete the Middle Node of a Linked List
- ③ Merge two sorted Linked Lists.
- ④ Reverse A Linked List
- ⑤ Intersection Point of two Linked Lists
- ⑥ Floyd's Cycle Finding Algorithm (Hare & Tortoise)



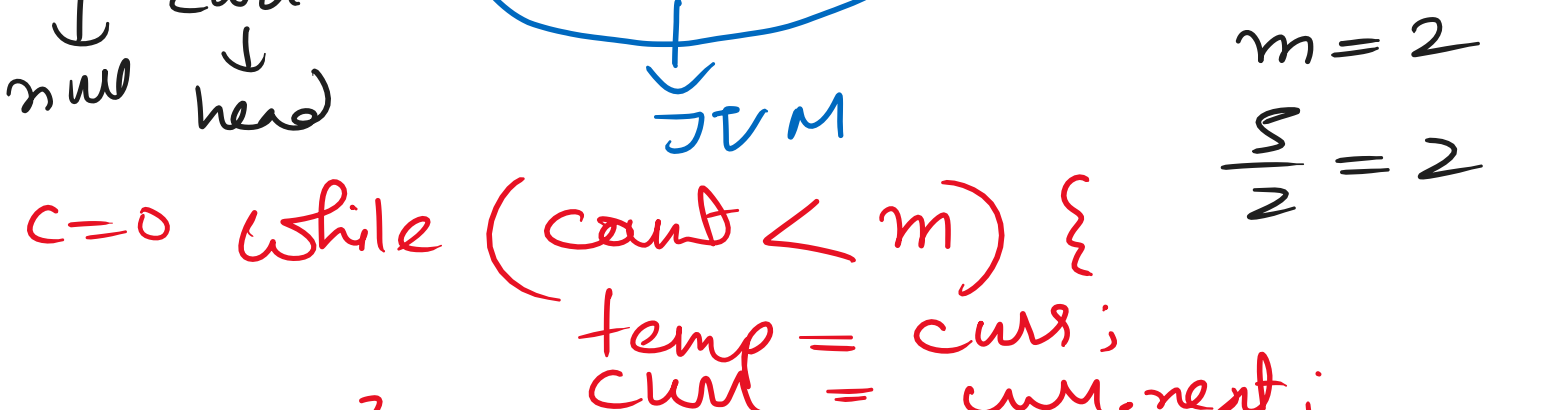
while (fast != null) {  
    slow = slow.next;  
    fast = fast.next.next;  
}



slow = head  
fast = head  
while (fast.next != null) {  
    slow = slow.next;  
    fast = fast.next.next;  
}



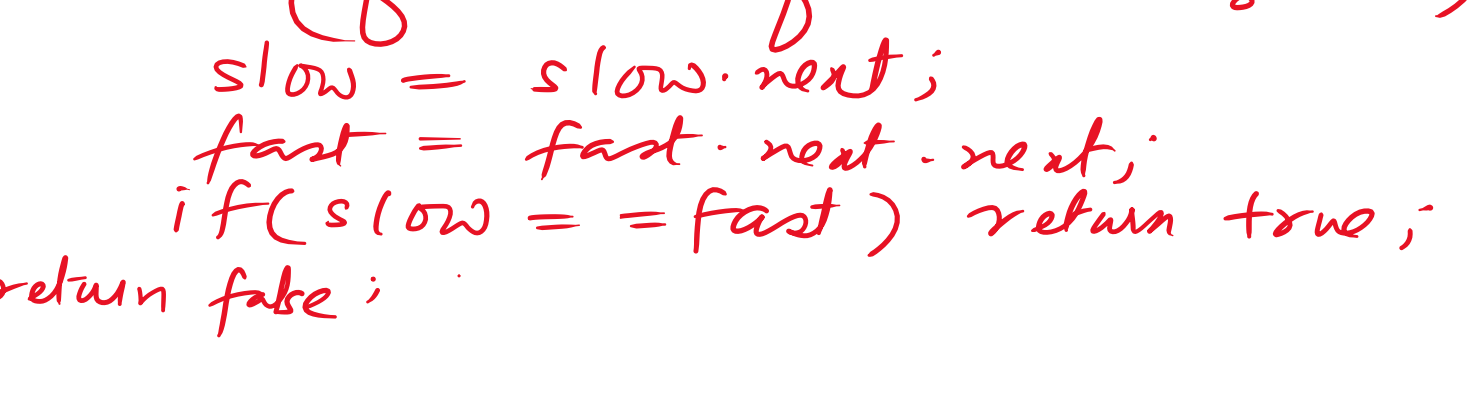
while (c < m) {  
    temp = temp.next;  
    c++;  
}



c=0 while (count < m) {  
    temp = curr;  
    curr = curr.next;  
    c++;  
}

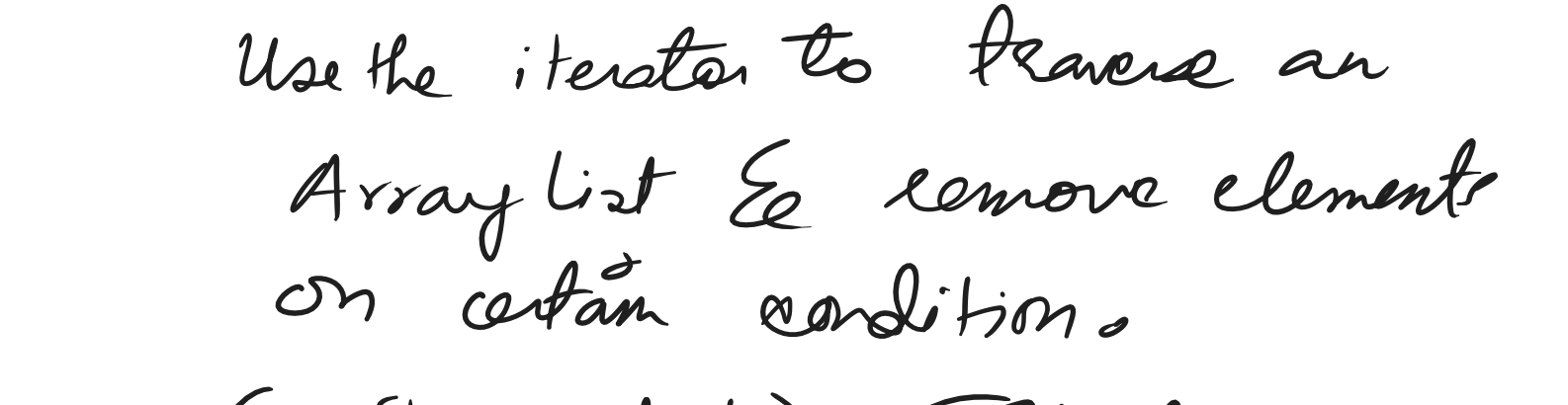
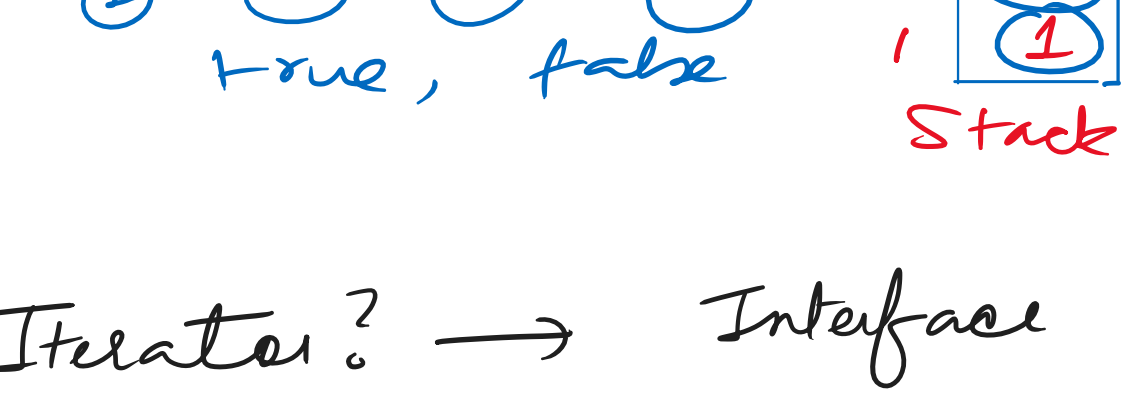
temp.next = curr.next;  
head;  
return head;

## Floyd's Hare & Tortoise Algorithm to find cycle in a Linked List:



while (fast && fast.next != null) {  
    slow = slow.next;  
    fast = fast.next.next;  
    if (slow == fast) return true;  
}

## \* Given a linked list, check whether it is a "Palindrome" or not.



Stack LIFO

## Iterator? → Interface

Use the iterator to traverse an ArrayList & remove elements on certain condition.

(java.util) Collections.

541k Coders Arcade