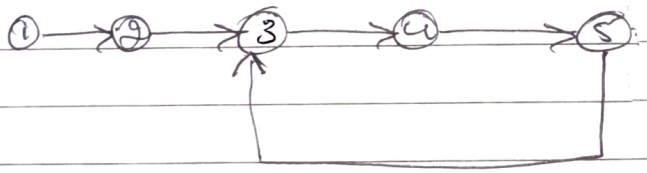
Venkatesha H D
Mithun M N
Ullas Gowda
G Shivabalan

Remove Loop Linked list

we need to detect the loop in the Linked list
if the loop exist then ~~remove~~ find the start of the loop
Remove the loop

=> Lets say we have the following linked list

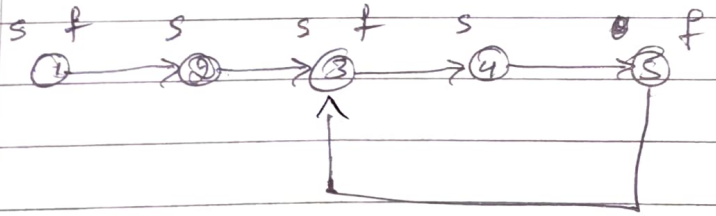


the last node 5 points back to node 3 creating a loop
!

=> Detect the loop (Floyd's Cycle Detection Algorithm)
*use two pointers:
* Slow pointer moves one step at a time
* fast pointer moves two step at a time

pointer Movement

step	slow at	fast at	Loop detected
1	1	1	NO
2	2	3	NO
3	3	5	NO
4	4	4	Yes



Venkatesha H.D
Nithun M N
Ulas Gowda
G Shivabalan

graphikaa
Date
Page

⇒ Find the Start of the Loop

* Set one pointer at head (ptr1 = head)
* Keep the other pointer at the meeting point (ptr2 = slo
* Move both pointers one step at time until they meet again.

pointer movement to start of the loop

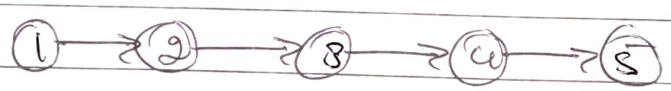
Step	ptr1 at	ptr2 at	Meeting point.
1	1	4	No
2	2	5	NO
3	3	3	Yes

Now we know Node 3 is the start of the loop.

⇒ Remove the loop

* start from the loop meeting point (ptr2) and find the last node before looping back (5).
* Break the loop by setting 5.next = None

Final list



python Code

```
def detect_and_remove_loop(head):
    slow = fast = head  # initialize two pointers

    # Detect loop using Floyd's cycle detection algorithm
    while fast and fast.next:
        slow = slow.next        # Move one step
        fast = fast.next.next   # Move two step

        if slow == fast  # if they meet, a loop exists
            remove_loop(head, slow) # remove the loop
            return True             # Loop was found and removed
    return False # no loop found


def remove_loop(head, loop_node):
    ptr1 = head     # pointer to traverse from the head

    while True:
        ptr2 = loop_node # Start from loop node

        # Find the node whose next pointer points to the
          start of the loop
        while ptr2.next != node and ptr2.next != ptr1:
            ptr2.next = None
            return
        ptr1 = ptr1.next # move ptr1 one step forward
```

```
def check_no_loop(head):
    slow = fast = head.

    # check for a loop using Floyd's Cycle detection algorithm
    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next
        if slow == fast:
            return False    # Loop still exists.

    return True # no loop detected


arr = [1,2,3,4,5]
pos = 2 # Loop from last node to node at index 2 (1-based index)

print(process_linked_list(arr,pos)) # should return True

def process_linked_list(arr,pos):
    head = create_linked_list(arr, pos) # creat linked list
    detect_and_remove_loop(head) # detect and remove loop if
                                 present
    return check_no_loop(head) # verify that the list has no loop
```