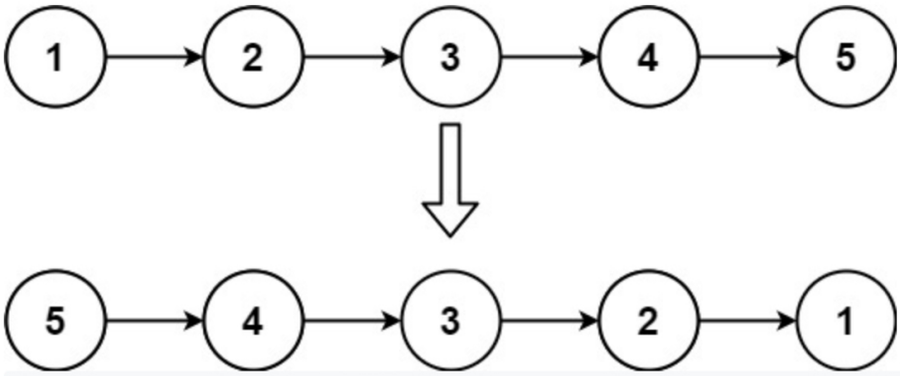
Reverse a linked list

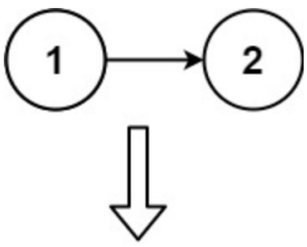
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:

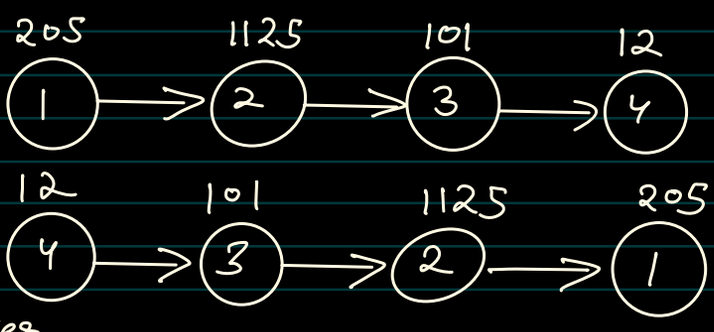



```
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
```

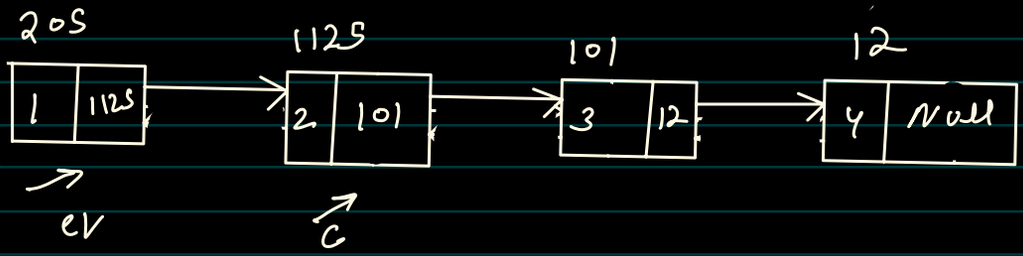
Example 2:



In Reversing the linked we have to reverse the Node not the data i.e



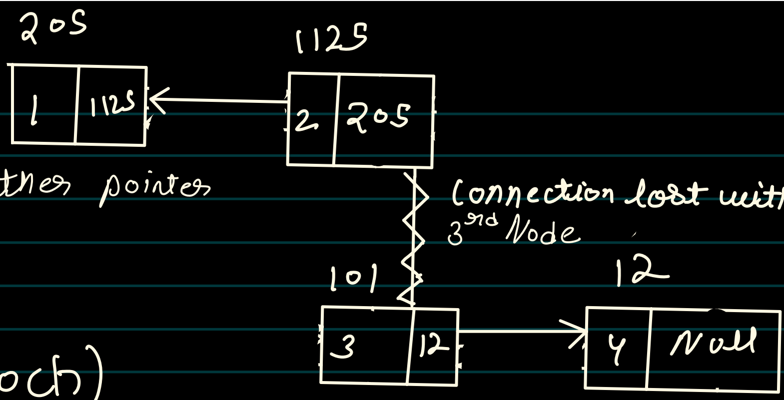
Two pointer



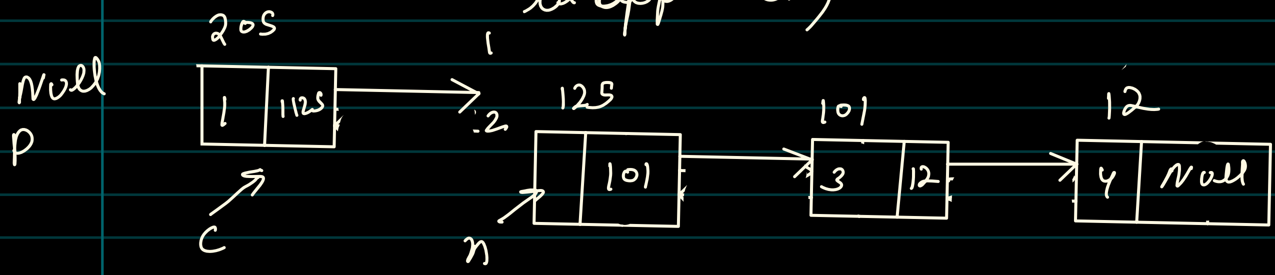
C->nes

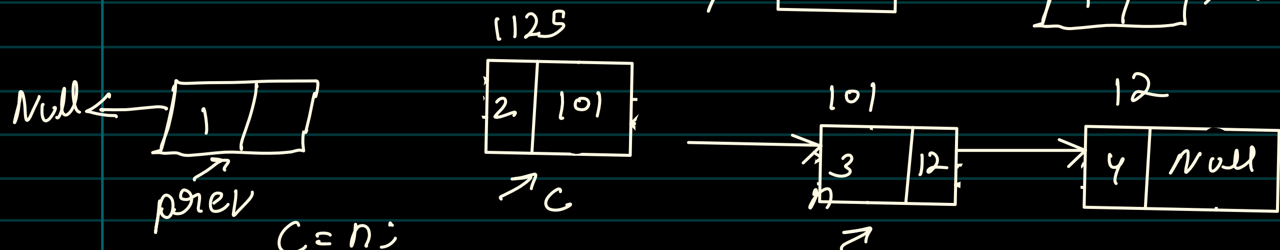
so we need t = prev;

to take another pointer



3 pointer approch (1st approch)





c = n;
n = n->next

while (c != Null)

c->next = prev;

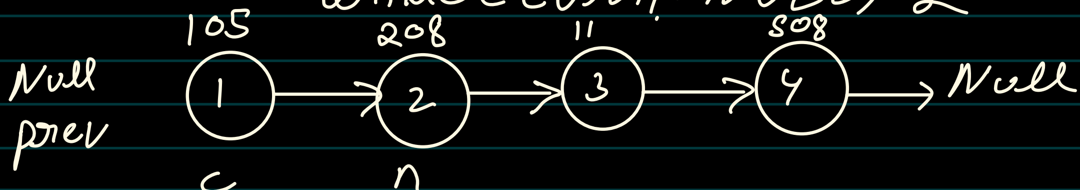
prev = c;
c = n;
if (n != NULL) n = n->next;

}

Working:

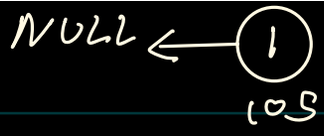
Prev = Null
Curr = head
n = head->next

while (curr != NULL) {

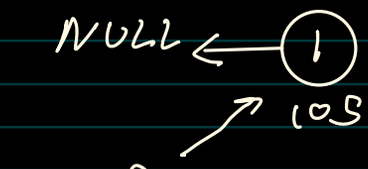


while (c != Null) True

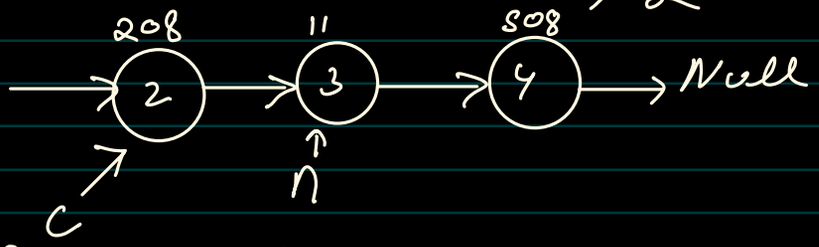
c->next = prev

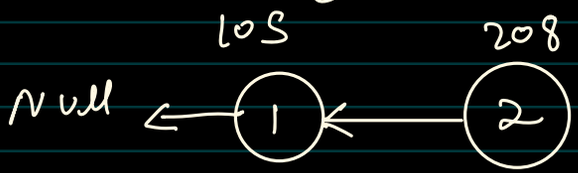


prev = c;



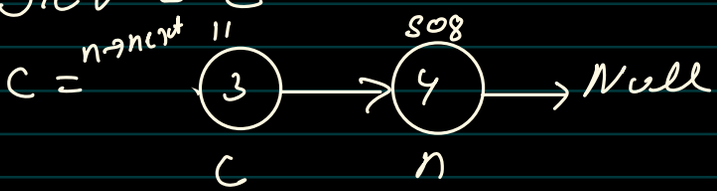
c = n → next & while (n != Null)



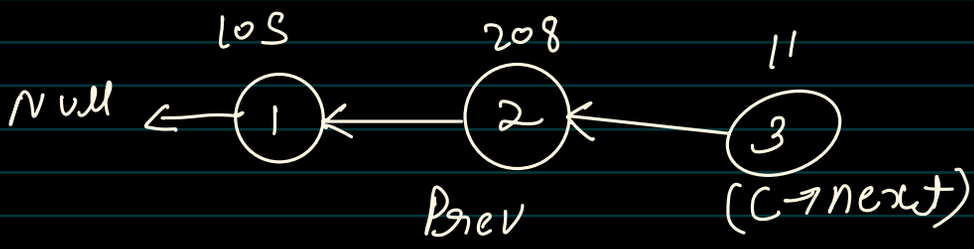


c → next = Prev

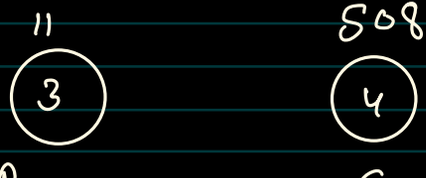
Prev = c



c → next = prev



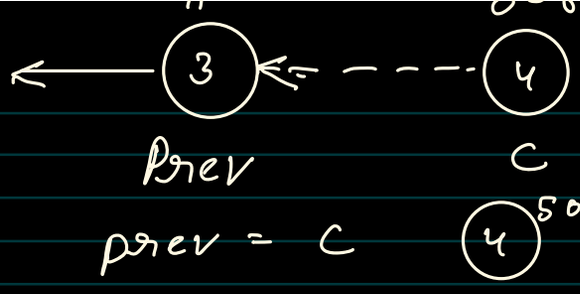
Prev = c



N = Null

c → next = Prev

11 508

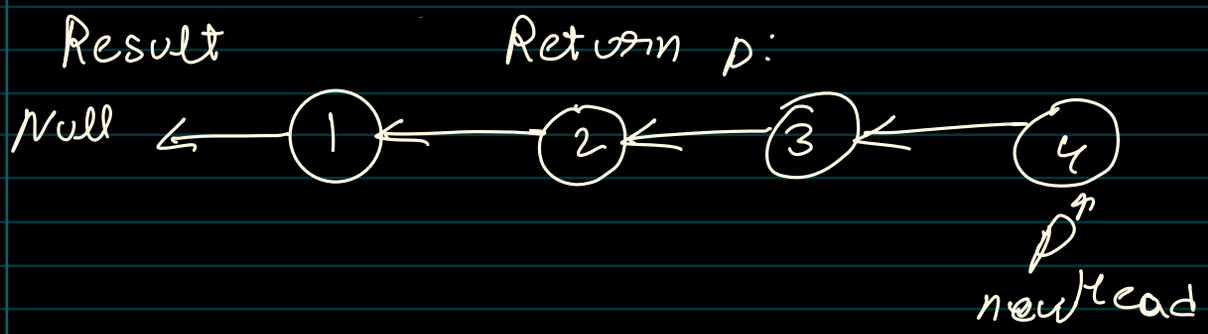


C = n → next

if (n → next != NULL) False

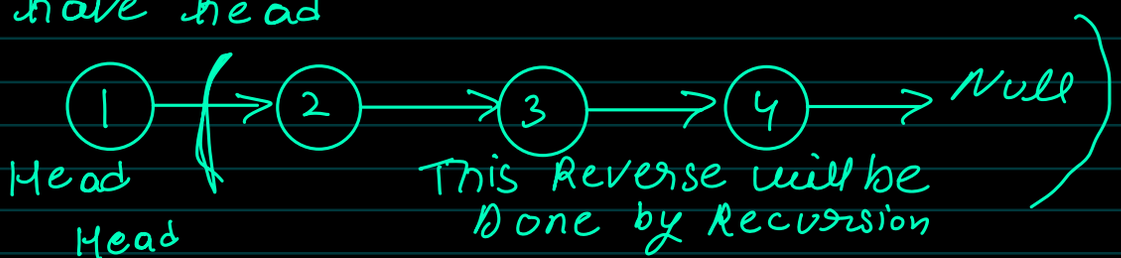
C (Null)

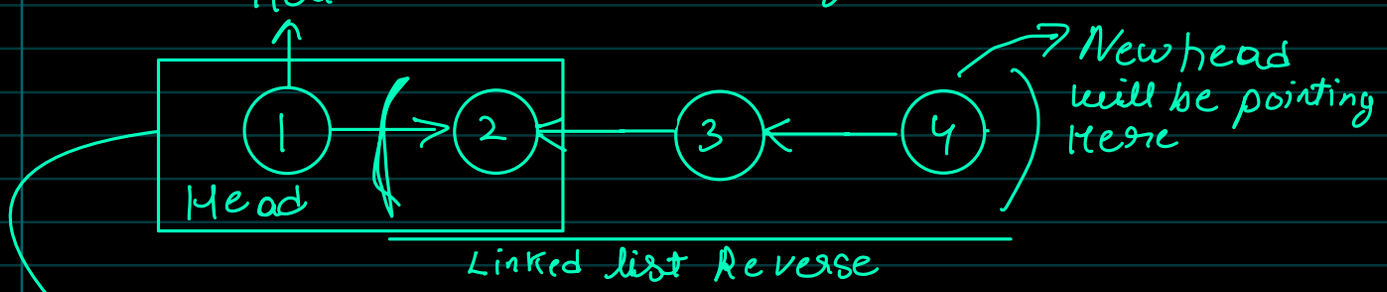
while (C != Null) False



2nd approch: (Recursive method)

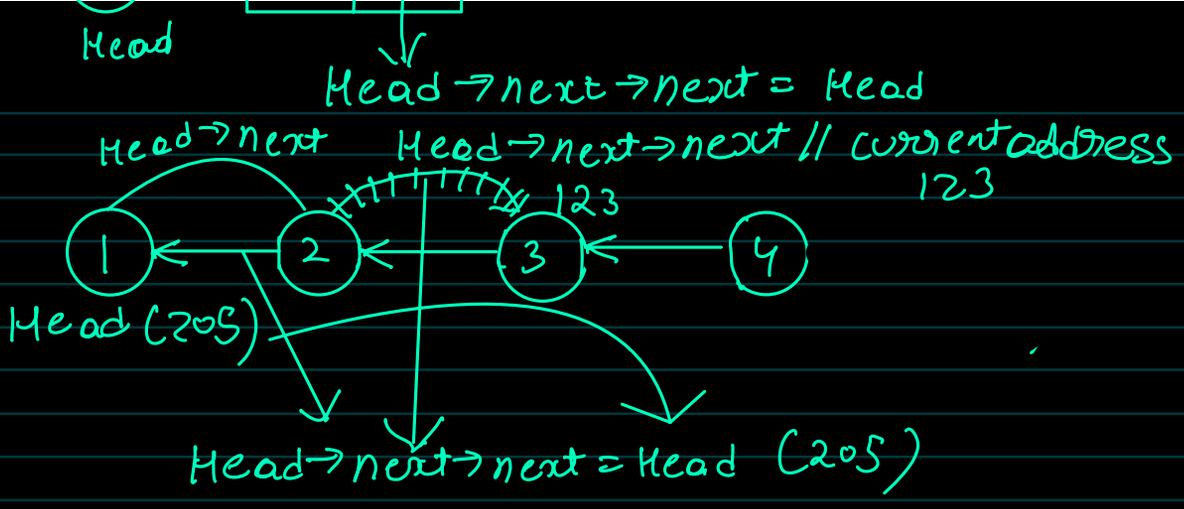
we have head





Our task is to point 2nd Node to 1st node and make First node pointing to NULL

Head → Next → Next




```cpp
class Solution {

private:

ListNode *reverse(ListNode *head)
{
    // Base Condition

    if(head -> next == NULL)
        return head ;

    // Ask recursion to return the Reversed Head

    ListNode *reverseHead = reverse(head -> next) ;

    // Reverse the Adjacent Nodes that is, Head & Head's Next Node

    head -> next -> next = head ;

    head -> next = NULL ;

    // Return the Reversed Head at the end

    return reverseHead ;
}
public:
    ListNode* reverseList(ListNode* head) {

    // If head is Equal to NULL, we don't have a LinkedList to reverse, so we directly return NUL

    if(head == NULL)
        return NULL ;

    // Otherwise we return the Reversed-Head given to us by the Recursive function reverse()

    return reverse(head) ;
}
};

/*
Time Complexity:  O(N)
Space Complexity: O(N) { Auxillary Stack Space of Recursion }
*/
```