Ques: 6

To check LL is a Palindrome or Not

head
↓
(1) → (2) → (3) → (2) → (1) → NULL

Left → Right
12321

Right → Left
12321

L to R | R to L
read same.
then LL is palindrome

Approach #1

→ normal LL. (we have normal LL)
→ reverse LL (we will reverse the given LL) → O(n)

Compare → T/F

T.C = O(n)

As we have reverse a LL an extra space is created for reverse LL

S.C = O(n)

same has palindrome else not a palindrome

Approach #2

↳ we have LL we will copy the LL data in an array.

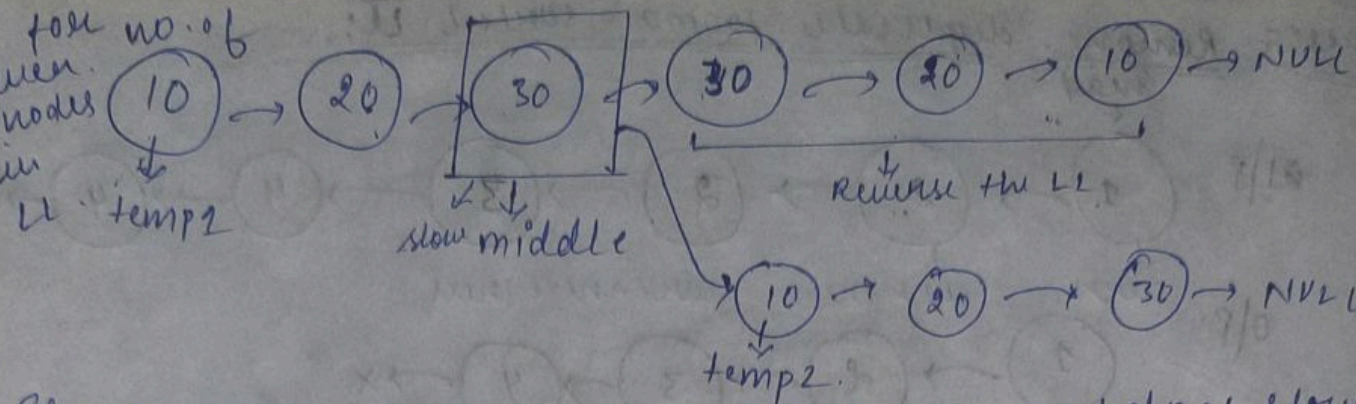
1	2	3	2	1

(s > e) → stop
to check palindrome.

T.C = O(n) as we have created extra array to store LL data ∴ S.C = O(n)

Approach #3:

(1) → (2) → (3) → (4) → (5)



Steps: (1) find the middle node of a LL with the help of slow/fast pointer

(2) Reverse the LL after the middle node.

(3) create temp1 ~~pointer~~ pointing to starting of nodes and temp2 pointing to first node of reverse LL & start comparing until temp2 become NULL if all are equal then it is palindrome.

Step (1) → T.C = $O(\frac{n}{2}) = O(n)$

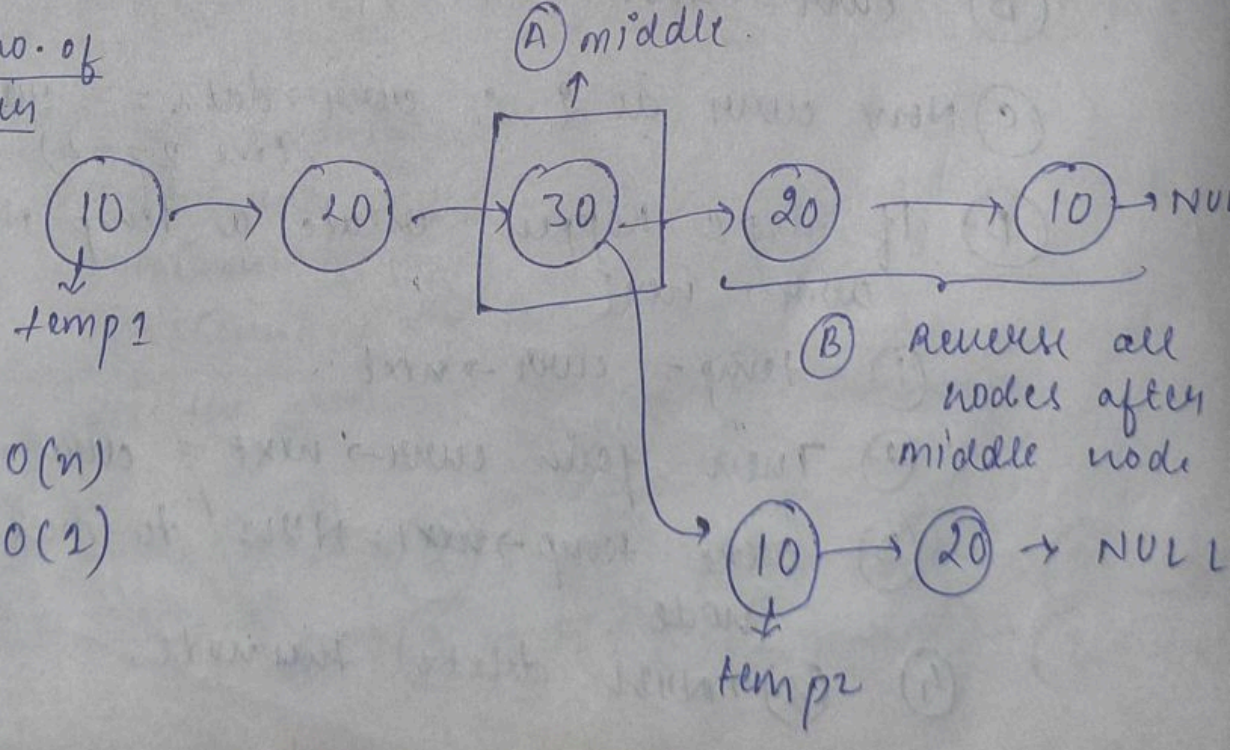
(2) → T.C = $O(\frac{n}{2})$ [Reverse half LL] i.e $O(n)$

(3) → T.C = $O(n/2)$ [comparing in 2 halves] i.e $O(n)$

Total T.C = $O(n)$

S.C = $O(1)$ [As we haven't created any extra space]

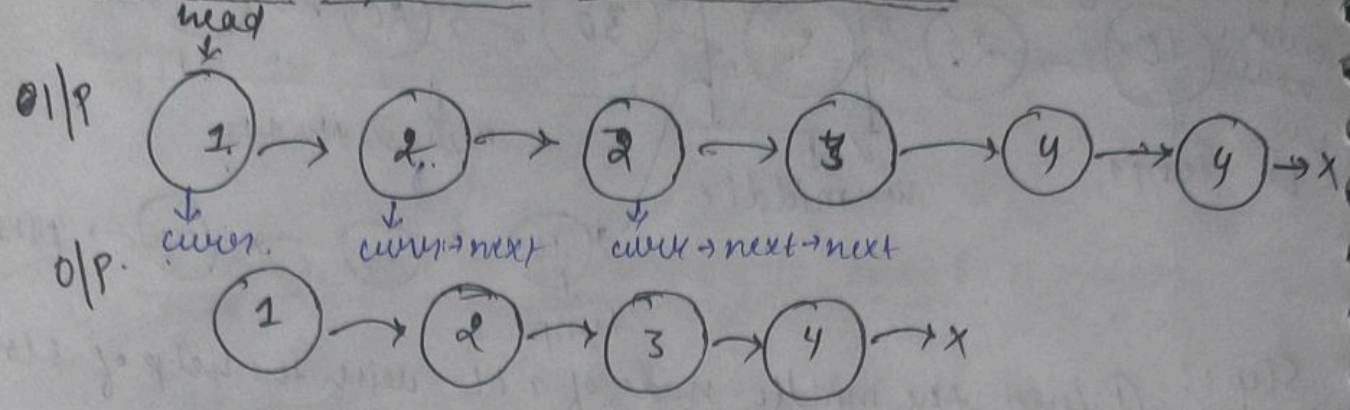
For odd no. of Nodes in LL



T.C = $O(n)$

S.C = $O(1)$

Ques:7 Remove duplicates from a sorted LL:



curr→data == curr→next→data

equal

when curr→data == curr→next→data

if both equal

~~delete curr next~~

curr = curr→next→next

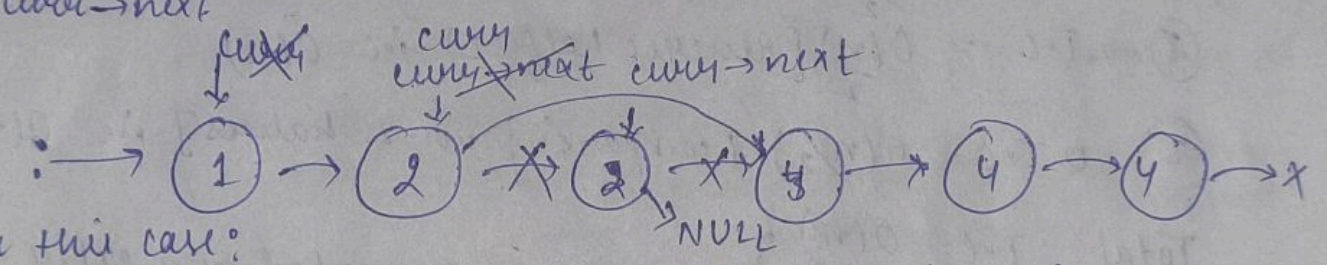
curr→next

not equal.

if curr→data != curr→next→data

move curr node ahead

curr = curr→next.



For this case:

(A) curr→data != curr→next→data (1 != 2)

(B) curr = curr→next

(C) Now curr is 2 ∴ curr→data == curr→next→data (i.e 2 == 2)

(D) If step C happen create a temp Node to store curr→next

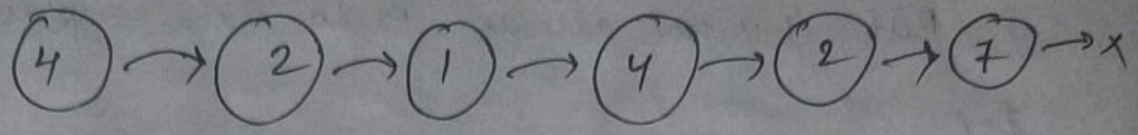
(1) temp = curr→next.

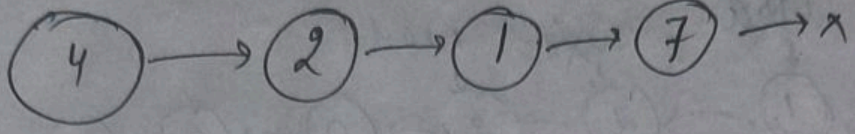
(2) Then join curr→next = curr→next→next.

(3) make temp→next = NULL to make it single node.

(4) (2)→NULL delete this node.

Ques:8 Remove duplicates from Unsorted list

I/P 

O/P 

Approaches

(1) Nested loop (one for each then traverse the entire nodes to check whether node is present or not)

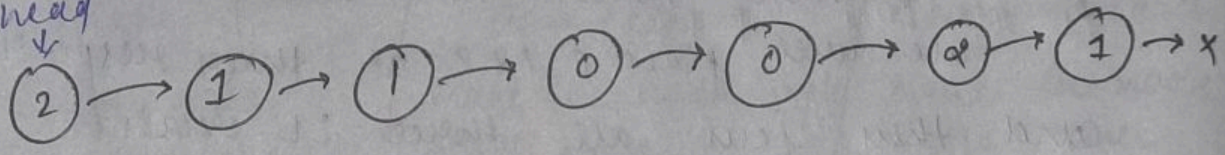
(2) map

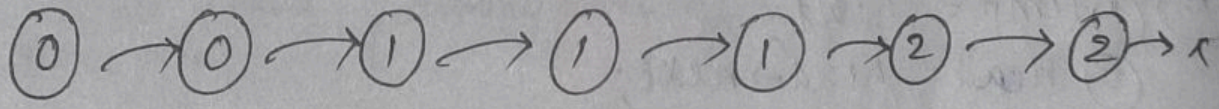
(3) first sort the unsorted LL then use the same method as remove duplicate from sorted LL

Ques:9 sorts 0's, 1's & 2's

V.V. Imp

head

I/P 

O/P 

Approach #1 use 3 variable-

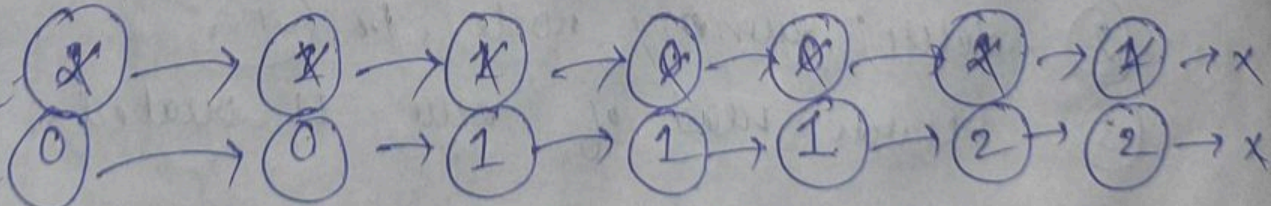
i.e ① zeros count = ~~0~~ X 2 so we have 2 0's, 3 1's & 2 2's

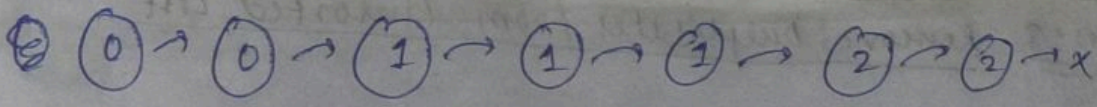
Ones Count = ~~0~~ X ~~2~~ 3

Twos Count = ~~0~~ X 2.

Now use the method of data replacement.

head

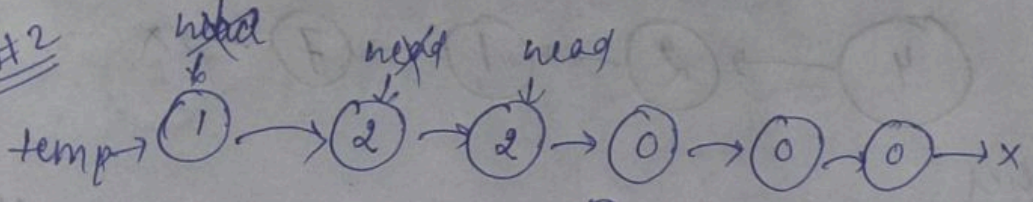
This is method of data replacement 

O/P 

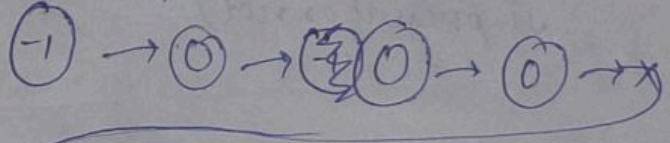
But it is not allowed to do data replacement.

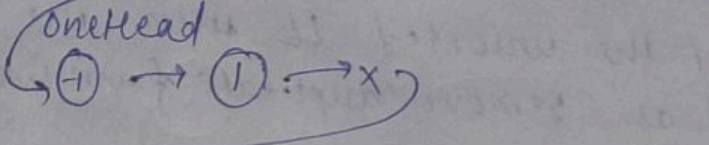
T.C

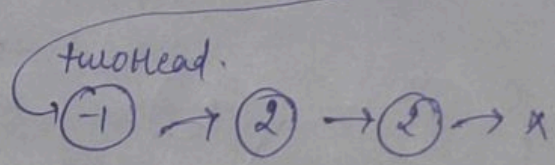
Approach #2



(A) create 3 different nodes with any values







In this approach rather than doing data replacement we will place 0, 1 & 2 at their respective nodes and they join all three LL created.

(B) (1) temp = head;
(2) head = head → next;
(C) temp → next = NULL

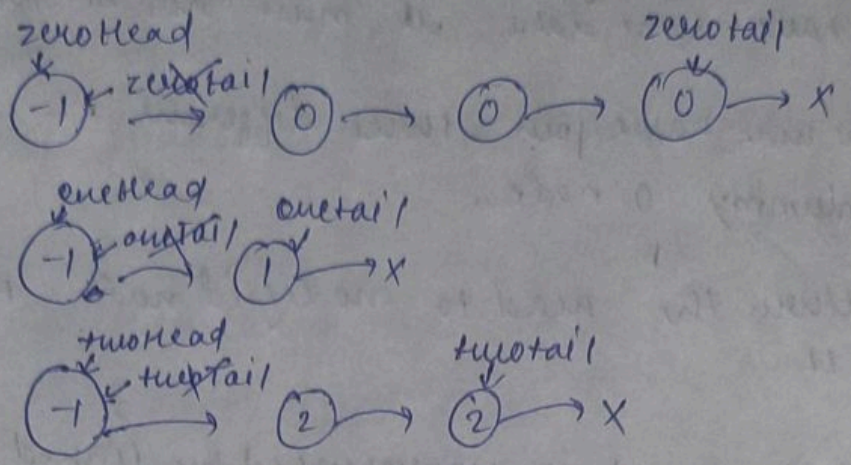
Now in step A : Once we have created all the nodes of original LL & placed them in 3 different nodes & the list formed.
we will join all 3 LL

(1) join 3 different LL
(2) delete dummy node i.e (-1)
(3) return head of new LL created.

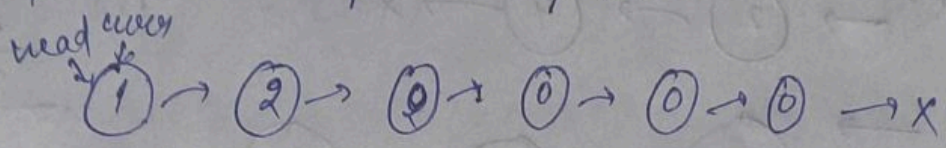
* Steps involved to sort 0's, 1's & 2's

(A) → Create dummy nodes which will add 0, 1, 2 and form a LL

(A)



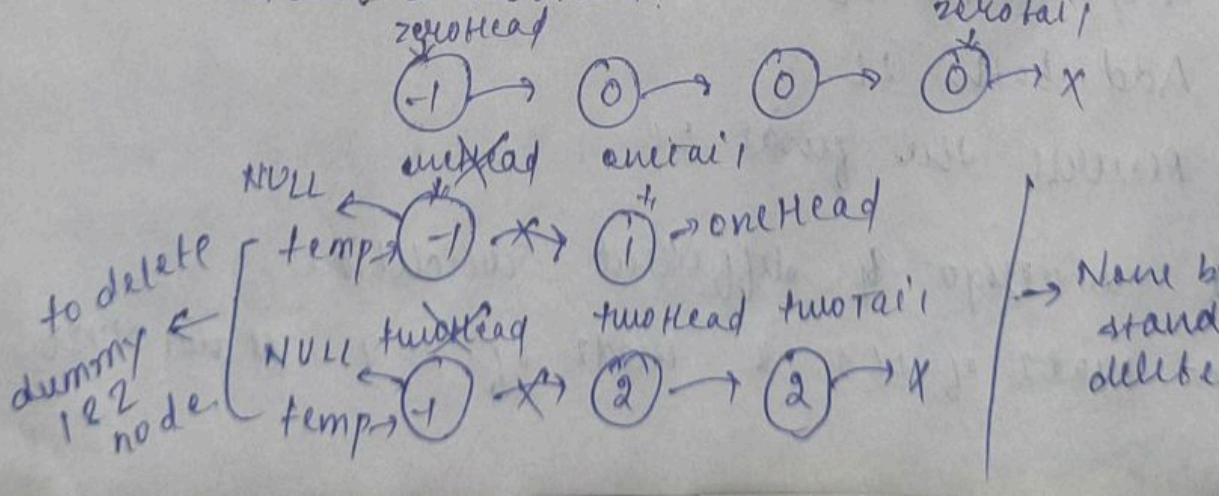
(B) to traverse through original LL & check if data is 0, 1 or 2. then place with resp. dummy node.



As we will traverse through entire Loop means use while loop until it becomes NULL.

curr → data = 0	curr → data = 1	curr → data = 2
Join the data with dummy node of 0. make the data single stand alone node	Make the node single stand alone & join the data with dummy node of 1.	Make the node single stand alone & join the data with dummy node of 2

(C) Now we need to join 3 different LL before that we will delete 0 dummy node of 1's & 2 i.e. oneHead & twoHead.



(D) Now we have delete dummy node 1 & 2. Now to join ~~the~~ ~~the~~ all three LL.

But we need to ensure that LL containing 1 & 2 must have some data it must not be NULL

(E) Now we have join three different LL. Delete the dummy 0 node.

(F) return the head to modified node. Then print the LL

Ques:10 Add 2 numbers represented by Linked List:

V.Imp

I/P: → (2) → (4) → X

i.e 24
234
258

→ (2) → (3) → (4) → X

Reverse the I/P LL

(4) → (2) → X

(4) → (3) → (2) → X

Now Add

O/P: (8) → (5) → (2) → X] → Reverse the final LL

Steps: (A) steps both LL

(B) Add both LL.

(C) Reverse the final LL

We will undergo 3 different condition

(1) when head1 of LL 1 & head2 of LL 2 is not NULL

(2) when head1 of LL1 is not NULL means head1 of LL is bigger than head2 LL

(3) when head2 of LL2 is not NULL means head2 of LL is bigger than head1 LL.

(4) when both head1 & head2 of LL is NULL.

means both LL is NULL then we will check for carry whether carry left of not

head1 head1 head1 head1 head2

→: (2) → (4) → (6) → (1) → (2) → X

head2 head2 head2

(9) → (5) → (6) → X

→ we will return this LL initial.

carry = 0 0 0 1 0

(head1 != NULL && head2 != NULL)

sum = 0 + 2 + 4 = 6

digit = 6

carry = 0

→

sum = 0 + 4 + 5 = 9

digit = 9

carry = 0

→

sum = 0 + 6 + 6 = 12

digit = 2 sum % 10

carry = 1 sum / 10

Resulted LL is

(6) → (9) → (2) → (2) → (2) → X

↓ Return resulted LL

(2) → (2) → (2) → (9) → (6) → X

↓ ans LL

→ as (head2 == NULL) we will go for when head1 != NULL

(head1 != NULL) Now carry is 1

sum = 1 + 2 = 3

digit = 3

carry = 0

sum = 0 + 2

digit = 3

carry = 0

Now head ≠ NULL & head2 is also NULL & carry is zero.