Linked List 1: Introduction

⇒ Issue with Array

int → 4 Bytes

↓

array → 5

↓

20 bytes

★ Array can't be created even if 20 Bytes of Free space is there because space is Discontinuous.

↓

Here, Linked List (LL) is saviour as it can be created at Discontinuous space.

★ LL → DS

LINKED LIST

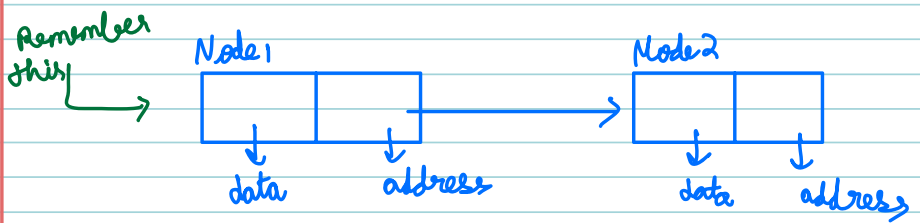
→ A Linear data structure that can utilize all the free memory.

→ We need not have continuous space to store nodes of a Linked List.

Real Life Example :- Waiting for match where Continuous Seats are not there.

↳ Need to remember the Sitting position of right person to remain in touch.

Q :- How will you store data in LL?

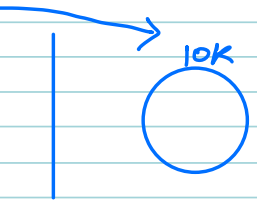


Q :- Is there any class which can store data & address of Next Node together?

↳ No, So we need to Create it.

⇒ Structure of Linked List

```
class Node {
    int data;
}

Node n = new Node();
Node temp = n;
```

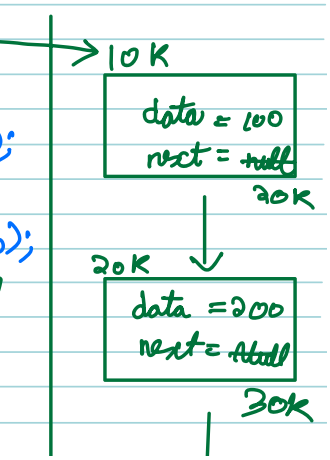


⇓

```
class Node {
    int data;
    Node next;
    Node(x){
        data = x;
        next = null;
    }
}
```

⇒ `Node t = new Node(100);`

⇒ `t.next = new Node(200);`



=> l.next.next = new Node(300);
30

30K
data = 300
Next = Null

Q If we need to pass LL, then how will you pass?

↳ Will pass head of LL

↓

address of first Node.

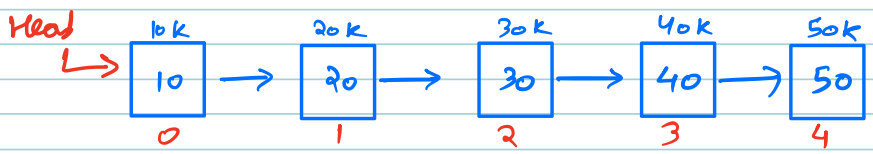
Quiz 1 :- Where will the "next" pointer of the last node point to ?

↳ NULL

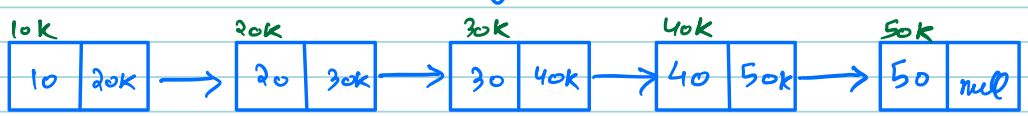
Quiz 2 :- From which node can we travel the entire Linked List ?

↳ First

=> Operation on Linked List



Right visualization



Question :- Access k^{th} index :-

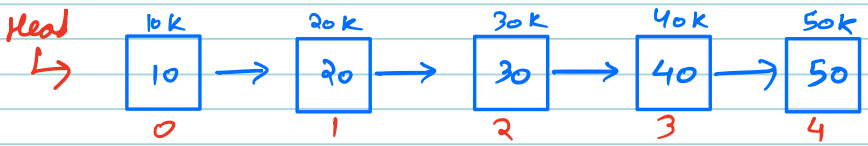
```
int  Kth index ( Node head (10k), int k (3)){
    Node temp = head
    for ( i = 0;  i < 3;  i++){
        temp = temp.next;
    }
    return temp.data ;
}
```

TC $\longrightarrow$ O(N)

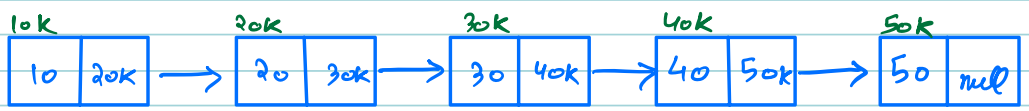
Q $\rightarrow$ why LL is asked ?

↳ It has Lot of edge Cases.

Question :- check for value x ?



Right visualization




```
bool check ( Node Head(10K), int x(40) ){

    Node temp = Head;

    while ( temp != null) {

        if (temp.data == x){
            return true;
        }

        temp = temp.next;
    }

    return False;
}
```

temp.next != null

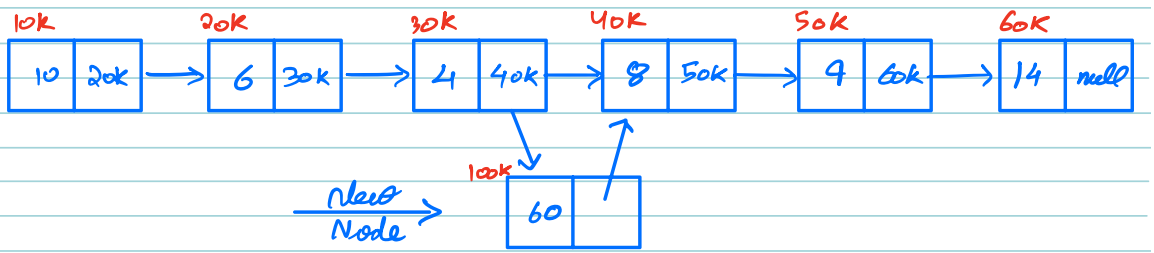
Quiz3 :- what is the time complexity to search any node in the Linked List ?

↳ O(N)

Quiz4 :- what is the time complexity to access the K^{th} element of the Linked List ? [index k is valid]

↳ O(K)

Question :- Insert a new node with Data.

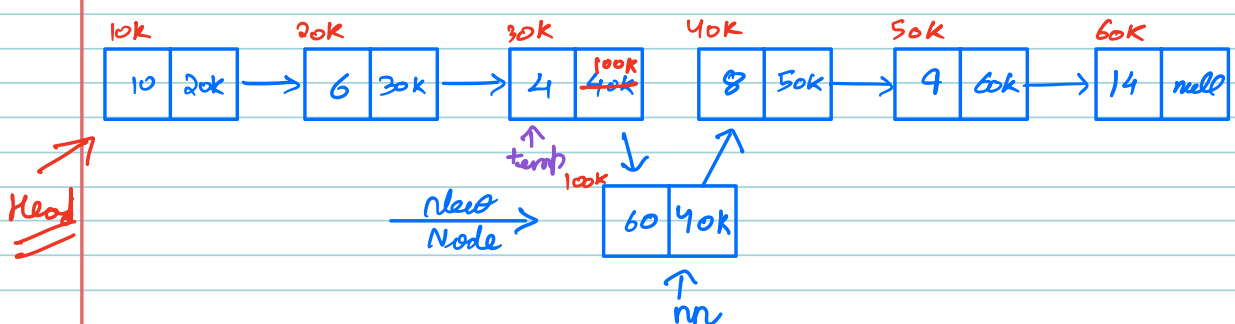


Quiz 5 :- Insert a new node with data 10 at index 2 in the given linked list.

Head → 1 → 6 → 7 → 9 → NULL. what will the correct ans after insertion.

↳ Head → 1 → 6 → 10 → 7 → 9 → Null

Ex value = 60 , Position = 3



Q Till where should go to start insertion ?

↳ 2^N position.

PSEUDO CODE

```
Node nn = new Node (value);
Node temp = head;
for (i = 0; i < Position-1; i++){
        temp = temp.next;
}
nn.next = temp.next
temp.next = nn;
```

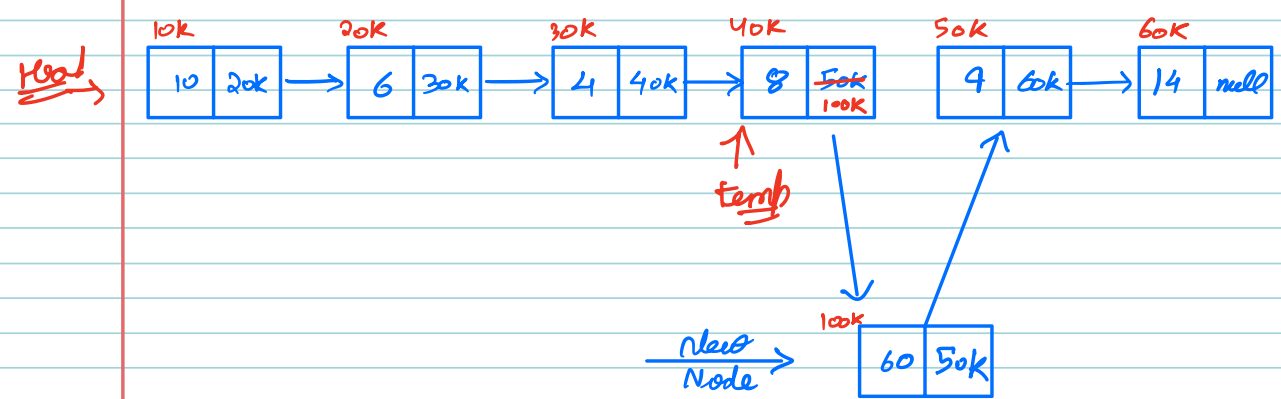
Problem ??

↳ Everything post 2nd index is lost.

Correction.

```
nn.next = temp.next;
```

Ex? :- Value = 60, Position = 4

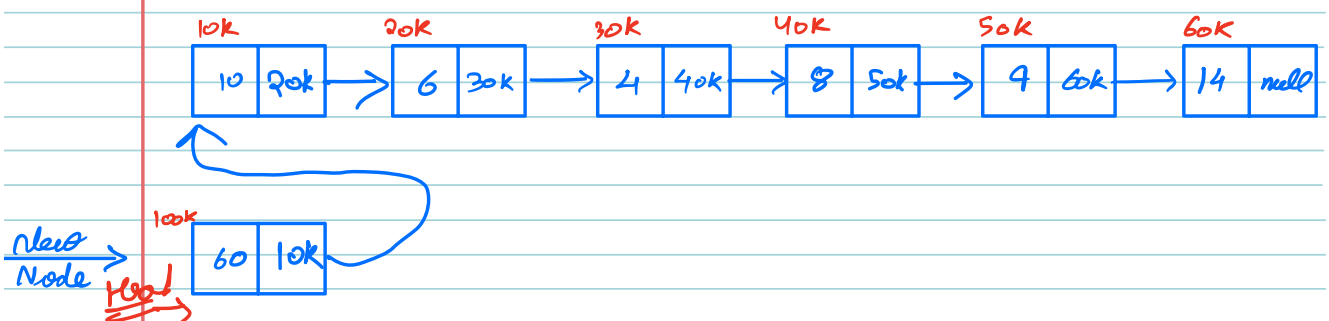


PSEUDO CODE

```
Node nn = new Node (value);
Node temp = head;
for (i=0; i < Position-1; i++){
        temp = temp.next;
}
nn.next = temp.next
temp.next = nn;
```

Edge Case

Ex2 :- Value = 60, Position = 0



PSEUDO CODE

```
Node nn = new Node (value);
Node temp = head;
for (i=0; i < Position-1; i++){
        temp = temp.next;
}
nn.next = temp.next
temp.next = nn;
```

Correction

```
if ( position == 0) {
    nn.next = head;
    head = nn;
}
```

PSEUDO CODE

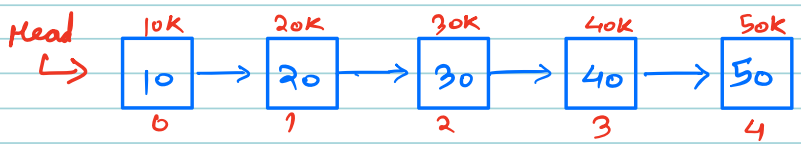
```
Node nn = new Node (value);
if ( position == 0) {
    nn.next = head;
    head = nn;
    return head;
}
Node temp = head;
for (i = 0; i < position-1 ; i++){
    temp = temp.next;
}
nn.next = temp.next
temp.next = nn;

return head;
```

Q :- when could be needed to return head ?

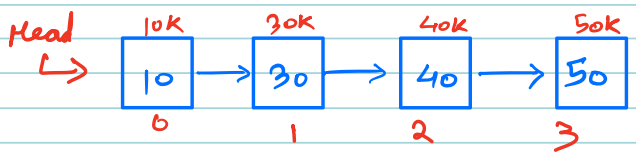
↳ change is not happening in place. So we need to pass the updated head to caller.

TC → O(N)

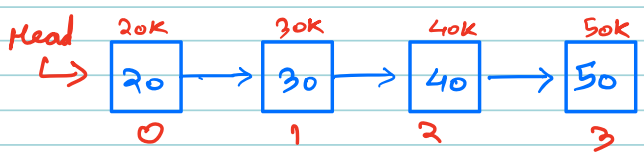
question :- Delete the first occurrence of value X in the given linked list. If element is not Present, leave as is.



Ex1 :- X = 20



Ex2 :- 10



Quiz 6 :- Delete the first occurrence of value X in the given linked list. If element is not present, leave as is.

Linked list :- 5 → 4 → 7 → 1 → NULL

X (to Delete) : 1

↳ 5 → 4 → 7 → NULL

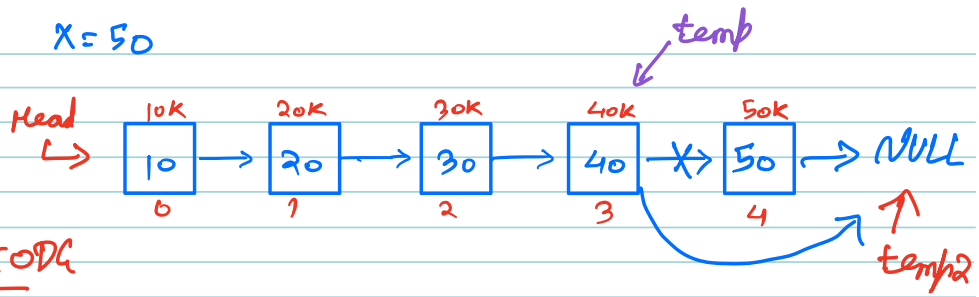
PSEUDO CODE

```
Node temp = head;
while (                ) {
    if ( temp.next.data == X) {
        temp2 = temp.next.next
        temp.next = temp2
    }
    temp = temp.next;
}
```

Q How to Delete when found element on next?

temp.next = temp.next.next

Ex :- X = 50

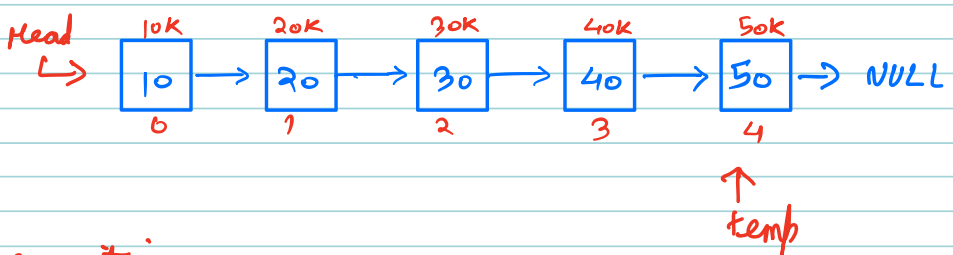


PSEUDO CODE

```
Node temp = head;
while (temp.next != null) {
    if ( temp.next.data == X) {
        temp2 = temp.next.next
        temp.next = temp2
    }
    temp = temp.next;
}
```

Q what will be the condition of while loop ?

X = 500

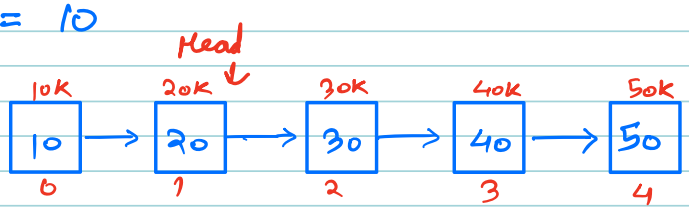


Condition

(temp.next != Null)

Edge Case

X = 10



Problem:- We are starting Comparision from 2nd Node.

Eg:-

```
if ( head. data == X){
    head = head. next
    return head;
}
```

PSEUDO CODE

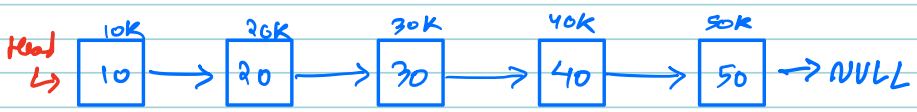
```
Node delete ( Node head, int X){
    if ( head. data == X){
        head = head. next
        return head;
    }
    Node temp = head;
    while (temp. next != null) {
        if ( temp. next. data == X) {
            temp2 = temp. next. next
            temp. next = temp2
        }
        temp = temp. next;
    }
```

return head;

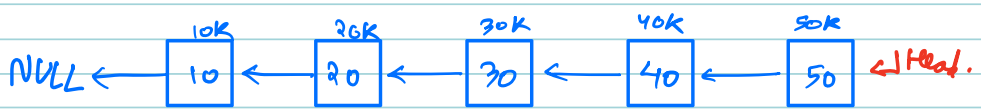
TC → O(N)

Question :- Reverse a Linked List.

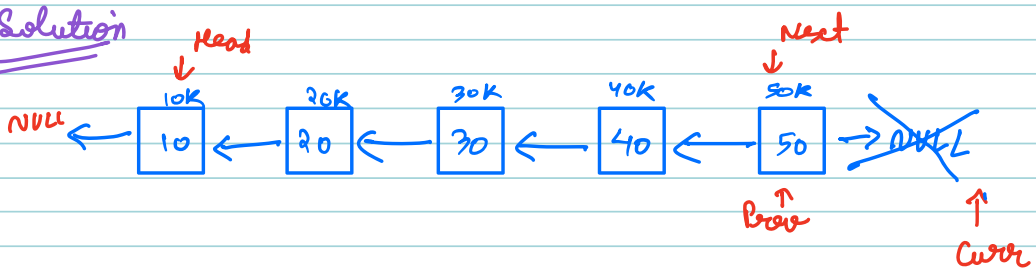
Ex :-



After Reversal



Solution



Way 1

curr.next = prev

↳ Problem ⇒ ahead LL is lost

Way 2 :-

```
next = curr.next;
curr.next = prev;
prev = curr;
curr = next;
```

Ques 7 :- Reverse the given Linked List :

5 → 6 → 7 → 8 → 9

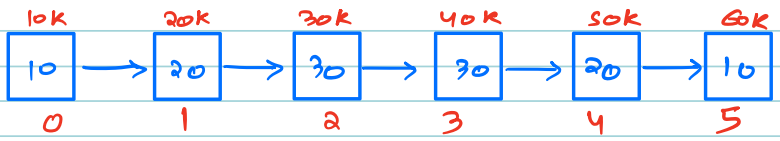
↳ NULL ← 5 ← 6 ← 7 ← 8 ← 9

PSEUDO CODE

```
Node reverse (Node head) {
    Node prev = NULL;
    Node curr = head;

    while (curr != null) {
        Node next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    head = prev;
    return head;
}
```

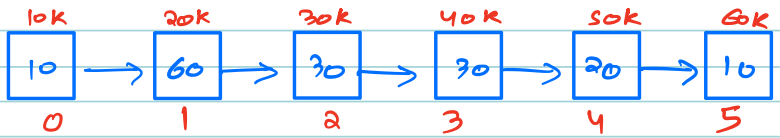
Question :- Given a Linked List, check if it is a palindrome.

Eg :-



↳ yes (True)

Eg2 :-



↳ No

Quiz :- Check the Given linked list is Palindrome or not.

Linked List :- Head → 1 → Null

↳ yes

IDEA :-

① Create a Copy of LL

② Reverse Copied LL

③ Compare Given & Reverse LL.

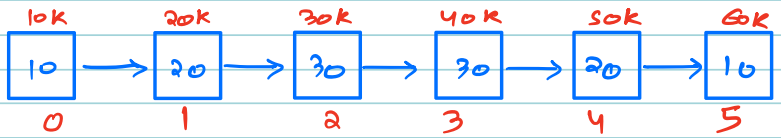
TC → O(N)

SC → O(N)

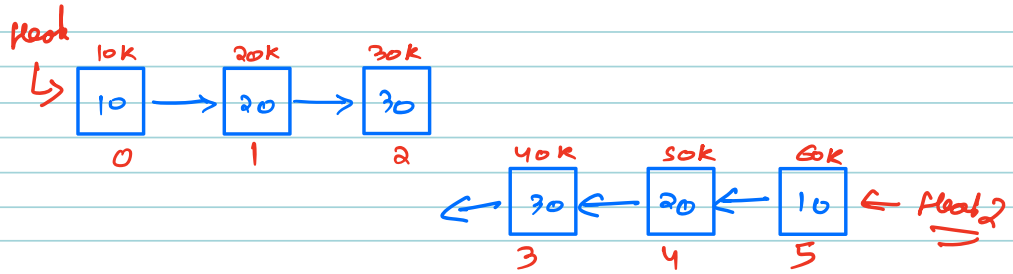
↳ OPTIMIZATION

⇒ How about breaking the LL into 2 parts & Reverse second half & compare?

Eg:-



⇒



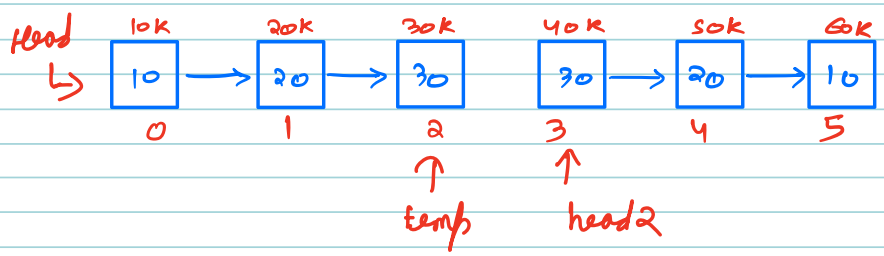
PSEUDO CODE

Step1:- Length of LL

```
n = 0
temp = head;
while (temp != null){
        temp = temp.next
        n++;
}
// n → 6
```

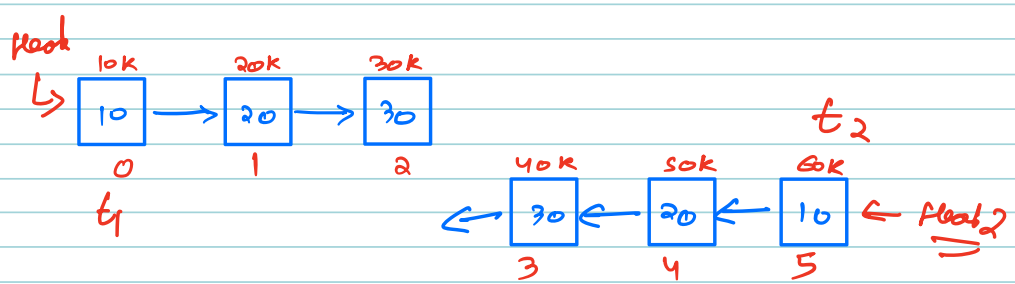
Step2 :- Goto half length & break;

```
int halfLength = n/2;
temp = head;
for (i=0; i< halfLength-1; i++){
    temp = temp.next;
}
Node head2 = temp.next;
temp.next = null;
```



Step3 :- Reverse Second half

```
Node head2 = reverse(head2);
```

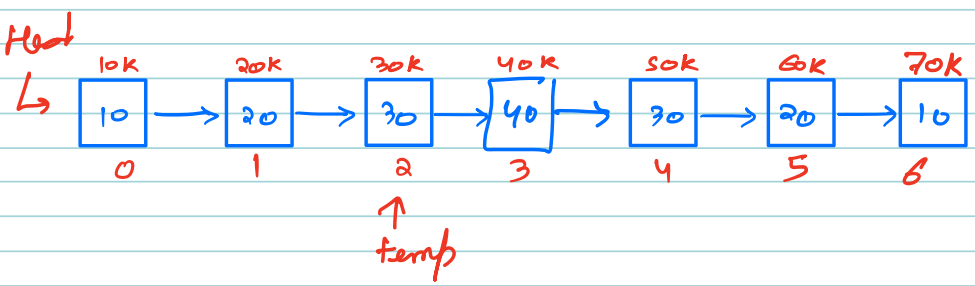


Step 4 :- Compare the Head & Head2 LL.

```
t1 = head1 , t2 = head2 ;
while ( t1 != null && t2 != null ) {
    if ( t1.data != t2.data ) {
        return False;
    }
    t1 = t1.next;
    t2 = t2.next;
}
return True;
```

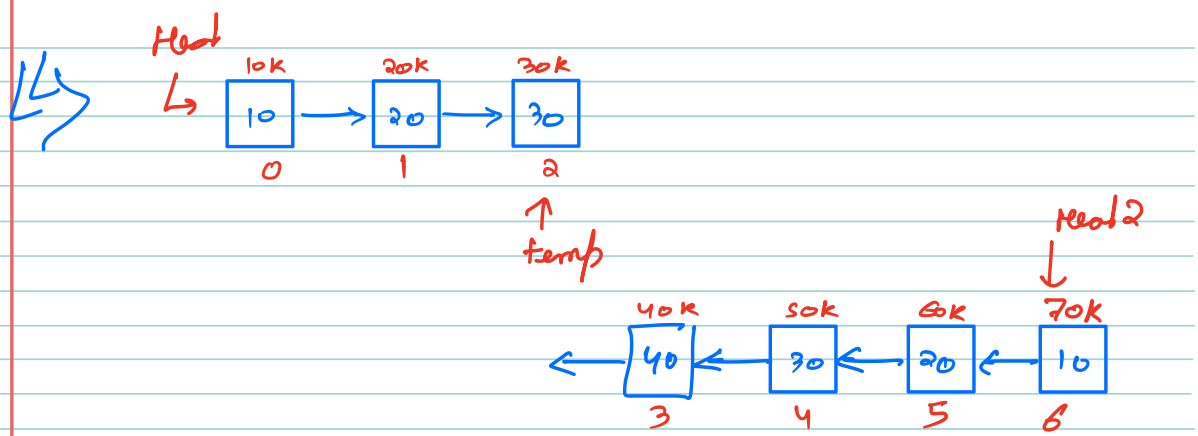
Dry Run for Odd Case

Eg:-



// Length = 7

// Half Length = 3


Head
10k
20k
30k
10
20
30
0
1
2
temp
Head2
40k
50k
60k
70k
40
30
20
10
3
4
5
6

TC → O(N)

SC → O(1)