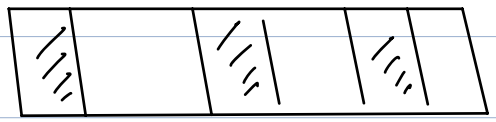
Issues with Array.

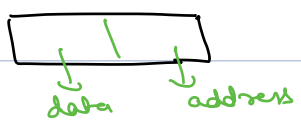


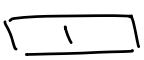
int → 4 Bytes
↓
array → 5
↓
20 Bytes.

LL → D.S. → not necessarily stores the data contiguously.

LL

- A linear data structure that can utilize all the free memory
- We need not have continuous space to store nodes of a Linked List.





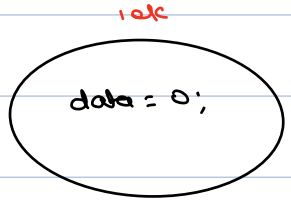

```
class Node
    int data;
```

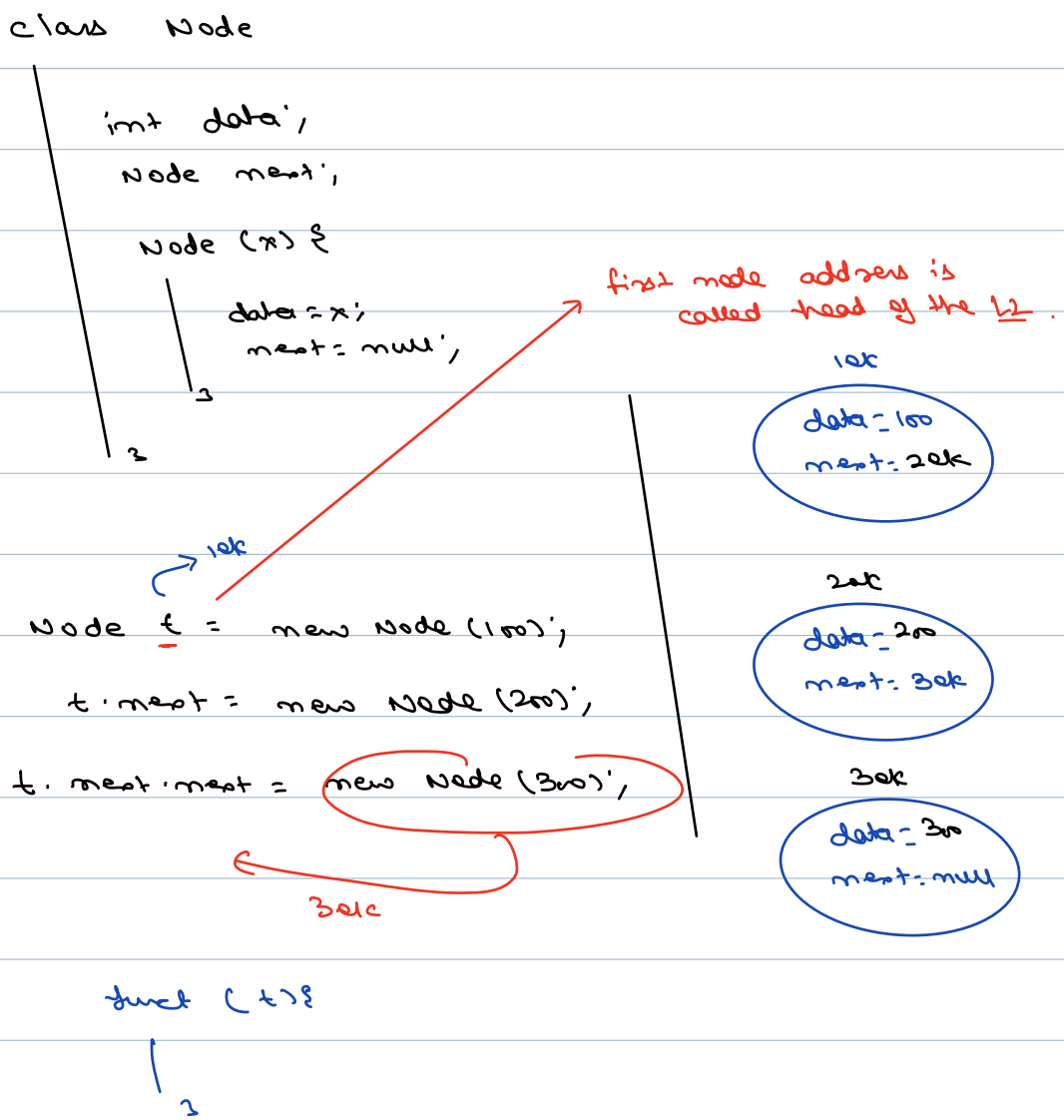
3

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

n → 10k.

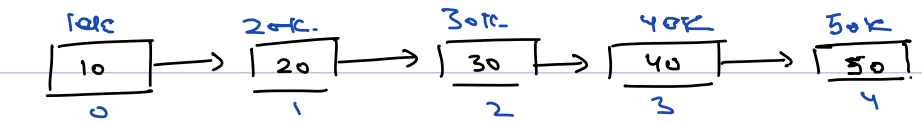
data type, variable.

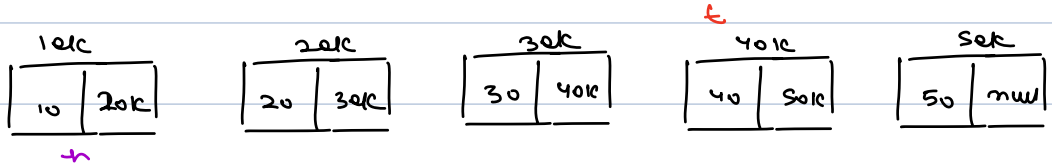



class Node
int data;
Node next;
Node (x) {
data = x;
next = null;
}
}
first node address is called head of the LL.
10k
data = 100
next = 20k
20k
data = 20
next = 30k
30k
data = 30
next = null
Node t = new Node (100);
t.next = new Node (20);
t.next.next = new Node (30);
10k
30k
fun (t) {
}

Operations on LL :-

k=3 → 40
k=1 → 20





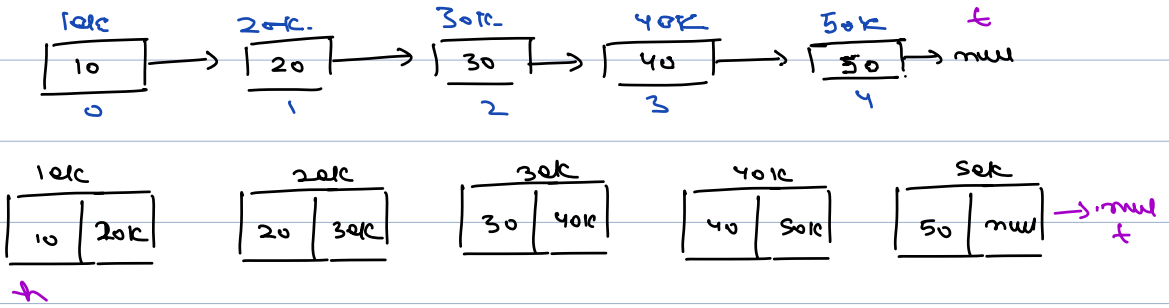
1) Access k^{th} idx:

T.C → O(n)

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

head → 10k
k → 3
temp → 10k, 20k, 30k, 40k
i = 0, 1, 2

2) Check for value x.

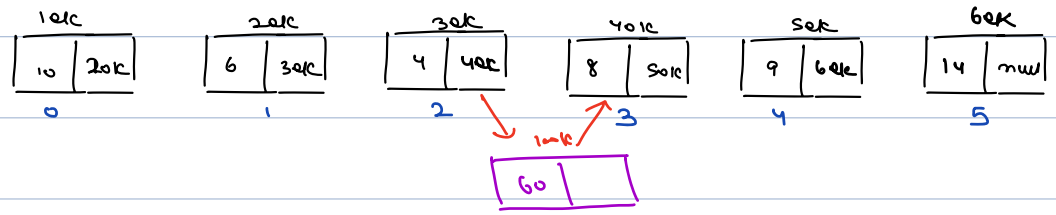


x = 30 → True
x = 100 → false.

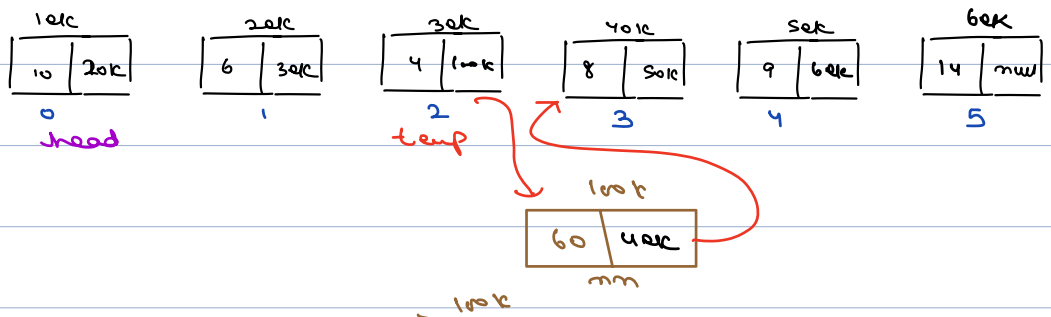
```
bool check ( Node head , int x ) {
    Node temp = head;
    while (temp != null) {
        if (temp.data == x) {
            return True;
        }
        temp = temp.next;
    }
    return false;
}
```

T.C → O(n)

Ques Insert a new Node with Data.



Ex1 V = 60 , P = 3



Node mm = new Node(v);

Node temp = head;

for (i=0; i<p-1; i++){

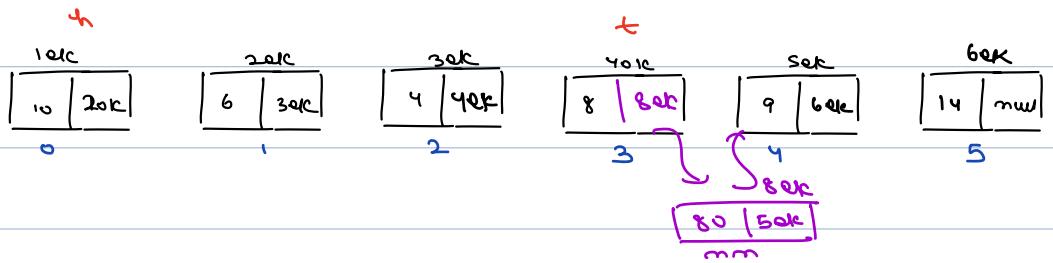
temp = temp.next

}

mm.next = temp.next

temp.next = mm;

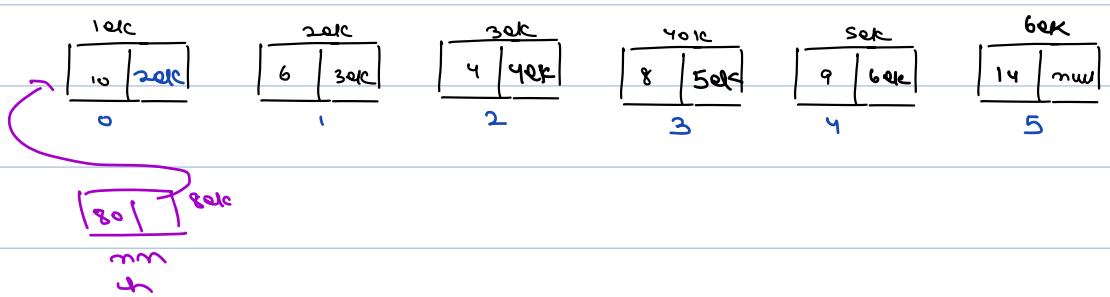
Ex1 V = 80, p = ~~4~~ 6




```
Node mm = new Node(v);

Node temp = head;
for (i = 0; i < p-1; i++){
    temp = temp.next
}
mm.next = temp.next ✓
temp.next = mm;
```

Edge case :-

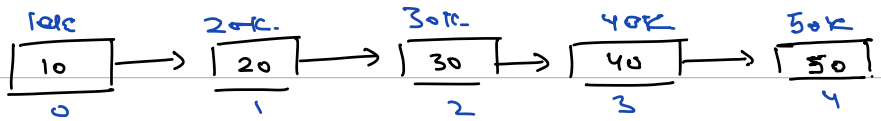
Ex1 V = 80, p = 0



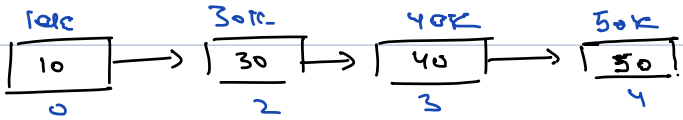
T.C → O(n).

```
Node insertAtK ( Node head , int v , int p ){

    Node nn = new Node (v)

    if ( p==0 ) {
        nn.next = h ;
        head = nn ;
        return head ;
    }
    Node temp = head ;
    for (i=0 ; i<p-1 ; i++){
        temp = temp.next
    }
    nn.next = temp.next ✓
    temp.next = nn ;
    return head ;
}
```

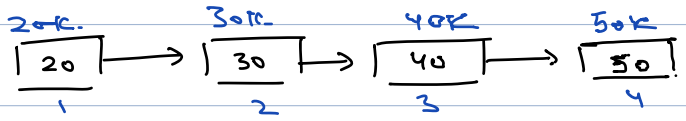
Ques Deletion in a LL.



e.g 1) x = 20

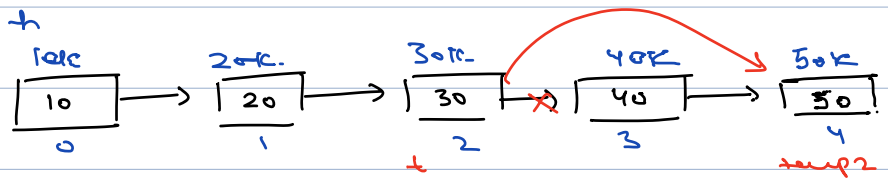


e.g 2) x = 10



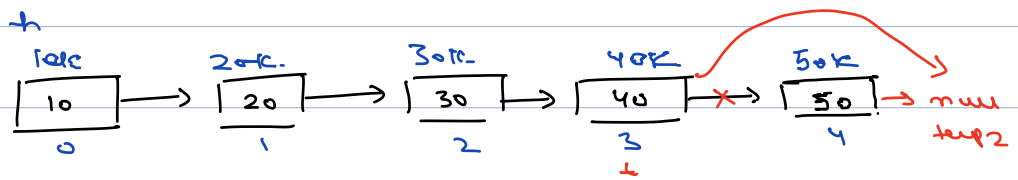
Soln

x = 40.



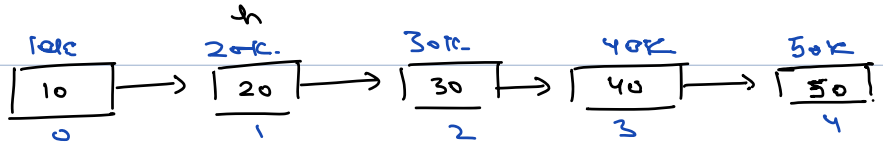

```
Node temp = head;
while (          ) {
    if (temp.next.data == x) {
        temp2 = temp.next.next;
        t.next = temp2;
        return head;
    }
    temp = temp.next;
}
```

x = 50.



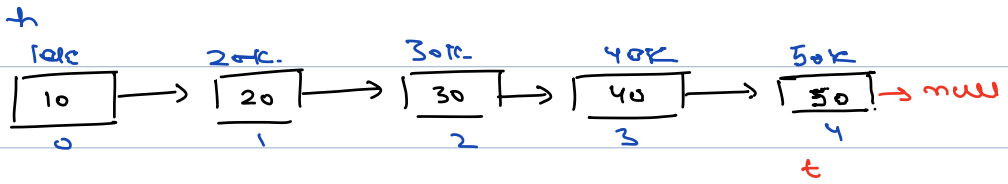

```
Node temp = head;
while (          ) {
    if (temp.next.data == x) {
        temp2 = temp.next.next;
        t.next = temp2;
        return head;
    }
    temp = temp.next;
}
```

x = 10.




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

x = 5.

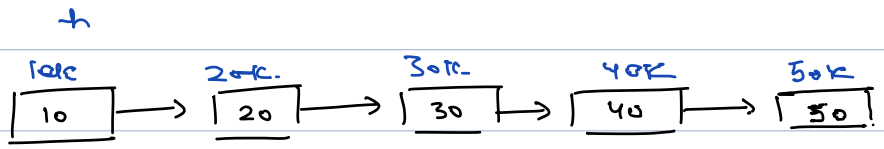



```
Node temp = head;
while (temp.next != null) {
    if (temp.next.data == x) {
        temp2 = temp.next.next;
        t.next = temp2;
        return head;
    }
    temp = temp.next;
}
```

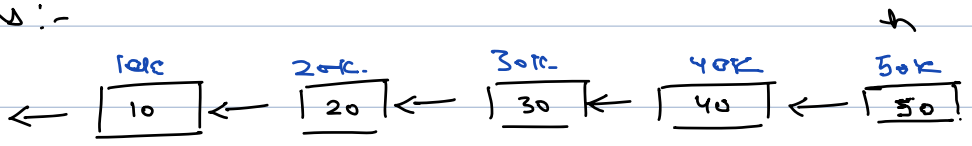
```
Node delete ( Node head, int x ) {

    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;
            t.next = temp2;
            return head;
        }
        temp = temp.next;
    }
    return head;
}
```

T.C -> O(n).

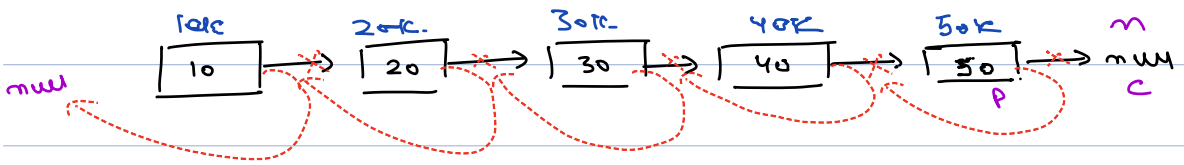
Reverse a LL



Ans :-



Soln

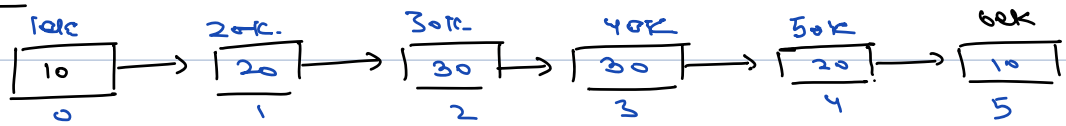


T.C → O(n)

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

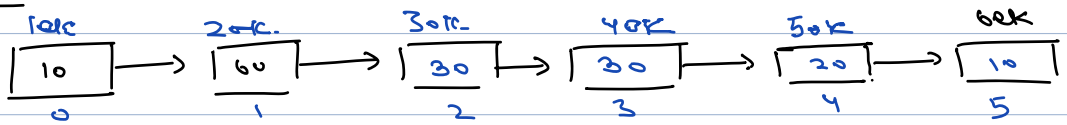
Ques Given a LL, check if it is palindrome.

e.g 1



return True

e.g 2



return false.

Soln :- 1). create a copy of LL.

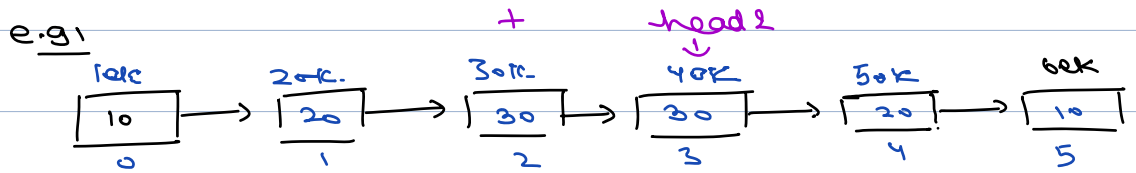
2) Reverse it

3) Compare

T.C → O(n)

S.C → O(n)

e.g:



Step-1 :- find len of h2

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

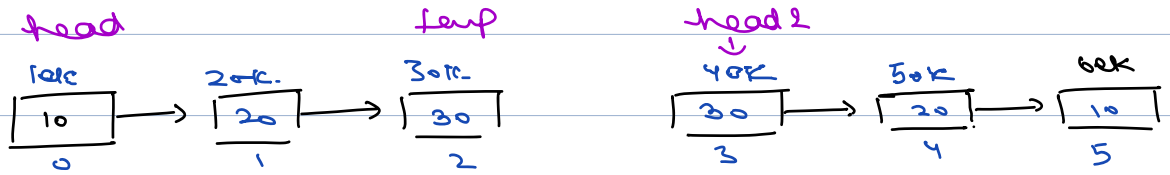
3

```
//n = 6.
int halflen = n/2
```

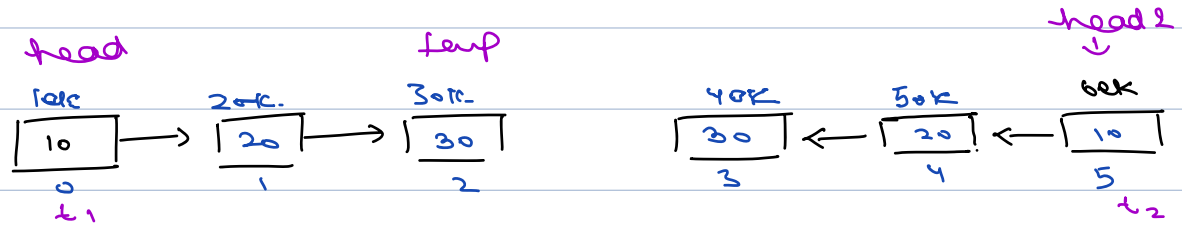
```
Node temp = head;
for (i = 0; i < halflen-1; i++) {
    temp = temp.next;
}
```

3

```
Node head2 = temp.next;
t.next = null;
```



head2 = reverse (head2);

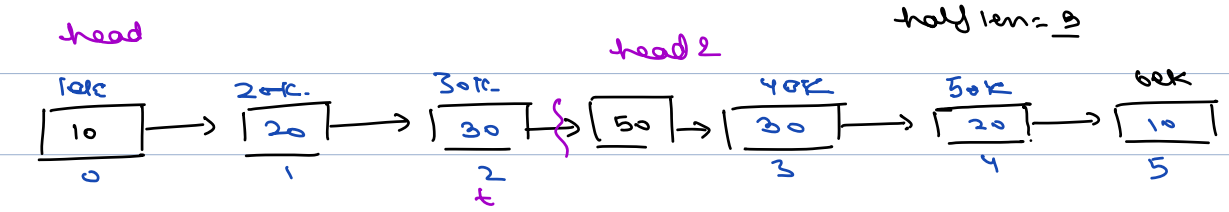


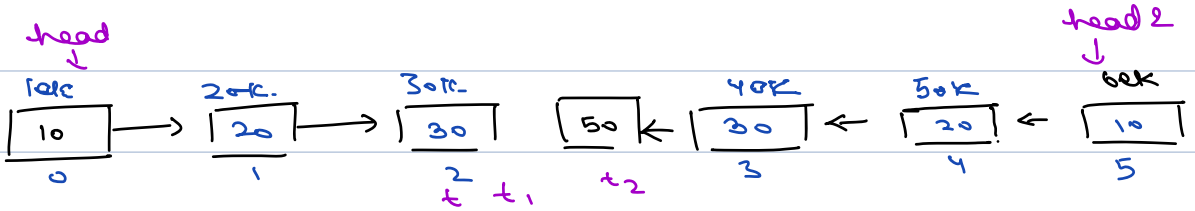

```
t1 = head;
t2 = head2;
while (t1 != null && t2 != null) {
    if (t1.data != t2.data) {
        return false;
    }
    t1 = t1.next;
    t2 = t2.next;
}
return True;
```

T.C → O(n)
S.C → O(1)

n = 7

half len = 3


head
head 2
10
20
30
50
30
20
10
0
1
2
3
4
5
t


head
head 2
10
20
30
50
30
20
10
0
1
2
3
4
5
t
t1
t2