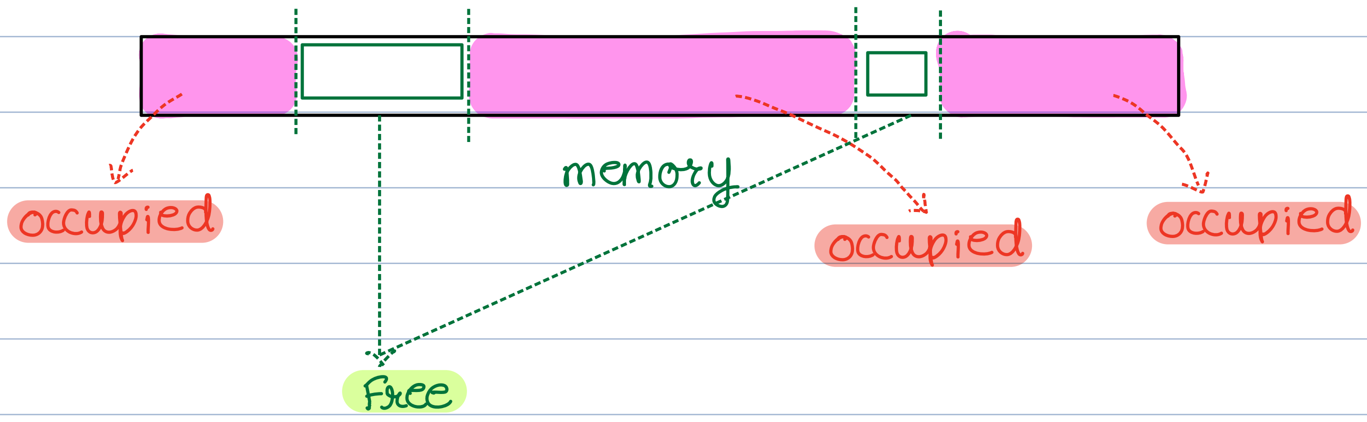
Linked List 1

Content

- $\longrightarrow$ Use of linked list over arrays.
- $\longrightarrow$ Basic functions { Access , Search }
- $\longrightarrow$ Insertion and deletion
- $\longrightarrow$ Reverse the linked list
- $\longrightarrow$ Palindrome list.



If you create an array will you be able to utilize all of the above free space.

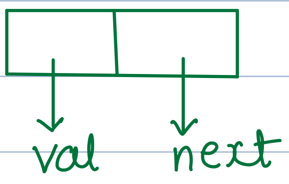
Array $\longrightarrow$ continuous block of memory.

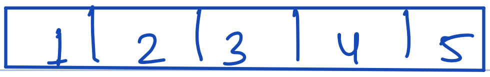
what is linked list ?

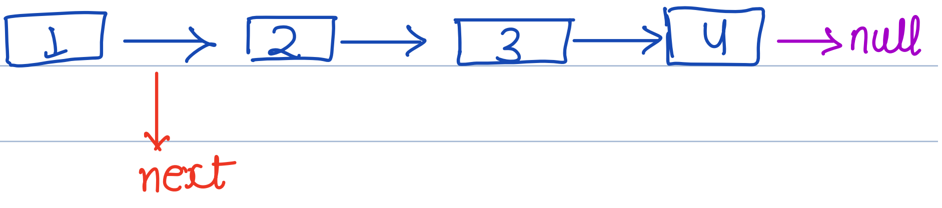
LL is a data structure which can occupy non continuous blocks of free memory.

Linked list $\longrightarrow$ Non continuous blocks of memory.

```
class Node {
    int val
    Node next

    Node (int x) {
        val = x
        next = null
    }
}
```







Operations

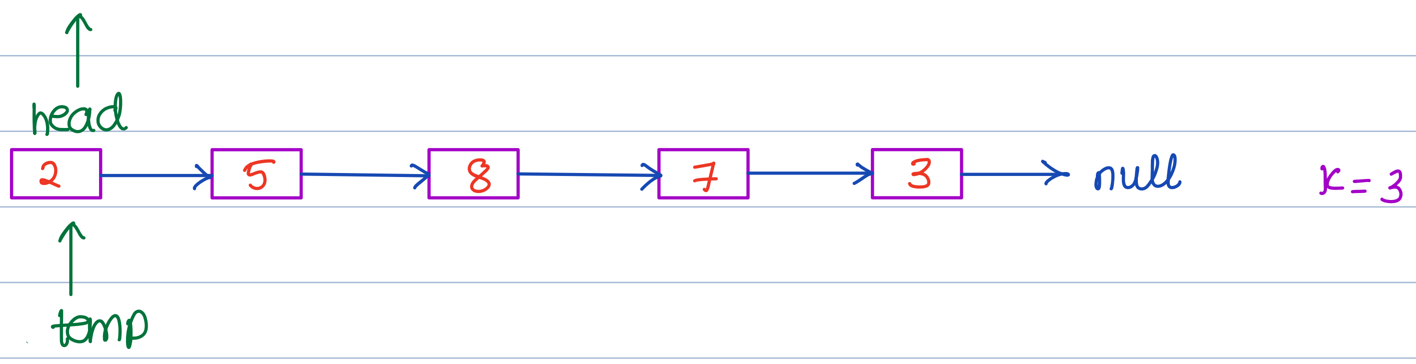
K is always valid

1> Access K^{th} element {K = 0 is the first element}

	0	1	2	3	4
A =	2	5	8	7	3

K = 3

A[K]



→ NOTE : Never move the head node during traversal, create a temp node and move it.

```
temp = head

for ( i = 0 ; i < K ; i++ ) {
        temp = temp . next        → to move temp to the next node
}

print ( temp.val )
```

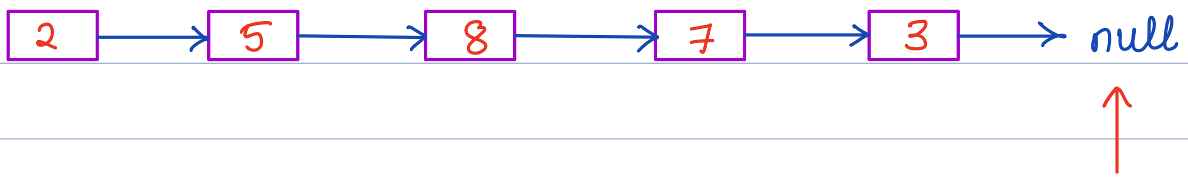
TC : O(K)

2> check for value X { searching }

Array → Linear search TC : O(N)
→ Binary search { sorted } TC : O(log N)

Linked List

X = 11



→ Linked List is empty

head = null → head.next → NPE

Null Pointer Exception

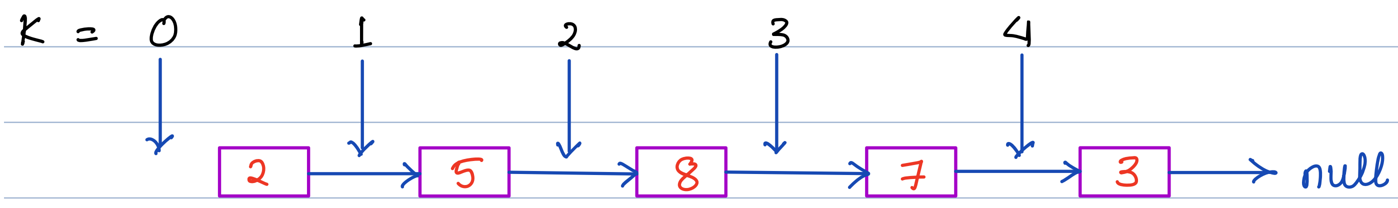
```
temp = head

while ( temp != null ) {
        if (temp.val == X) {
            return true.
        }
        temp = temp.next
}

return false
```

TC : O(N)

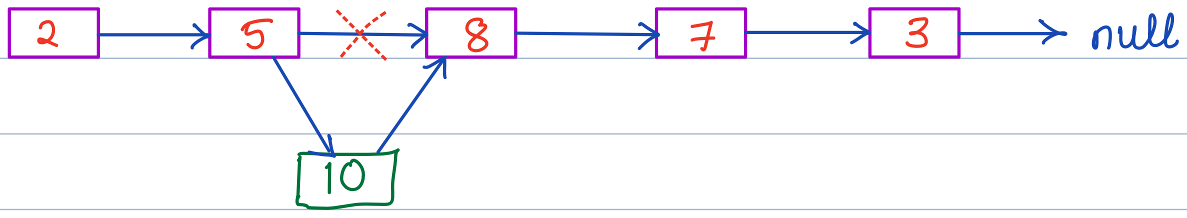
SC : O(1)

3> Insert a value X at k^th position {0-based}
in a linked list.

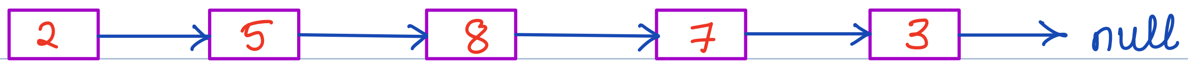
$0 <= K <= N$

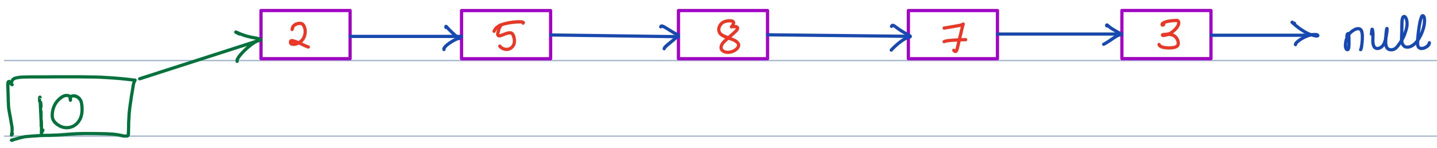


At k=2 insert X = 10

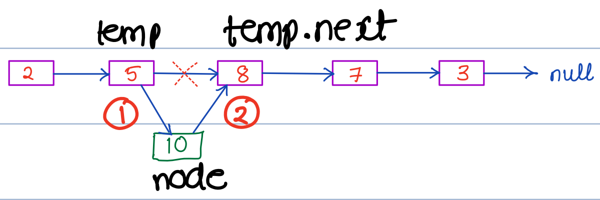


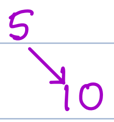
K = 0 , X = 10





```
node = new Node(x)    // Create a new
                         node with val x

if ( k == 0) {
        node.next = head
        head = node
}
else {

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

②       node.next = temp.next
①       temp.next = node

}

return head
```



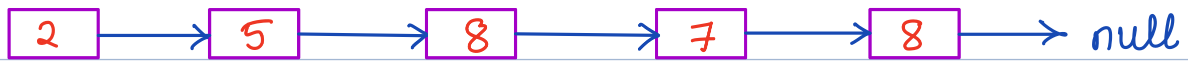


TC: O(k)

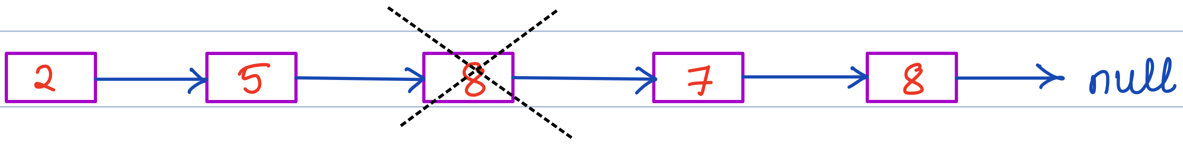
SC: O(1)

Q ⟶ Delete the first occurrence of value X in the given linked list. {If X is not present dont change}

Return the new head after deletion.



X = 8



2 ⟶ 5 ⟶ 7 ⟶ 8 ⟶ null

Cases

1> head is null

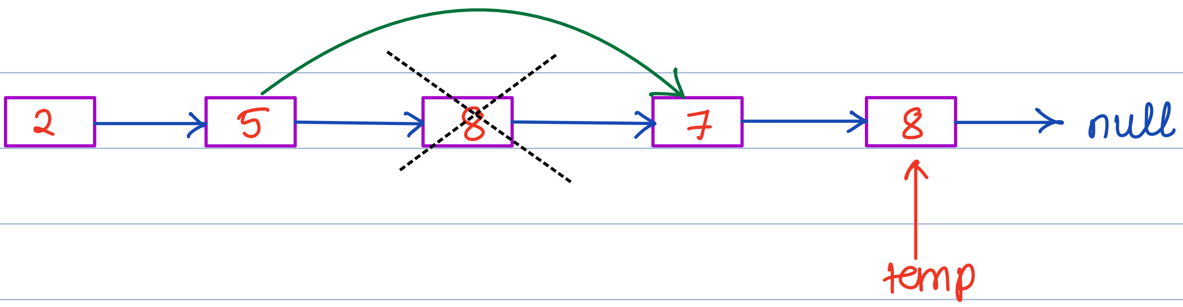
2> head.val == X

```
① if ( head == null ) {
       return head
   }
```

X = 2

2 ⟶ 5 ⟶ null

```
② if ( head.val == x ) {
       return head.next
   }
```



```
temp = head

while ( temp.next != null ) {
|
|      if ( temp.next.val == x ) {
|      |     temp.next = temp.next.next
|      |     return head
|      }
|
}      temp = temp.next

return head.
```

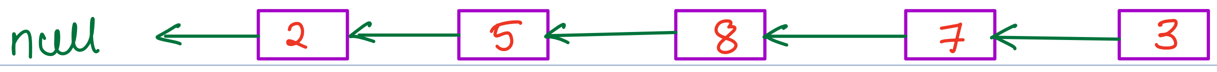
TC : O(N)

SC : O(1)

8:39 am

Reverse the Linked List

Donot change values, you need to swap pointers.

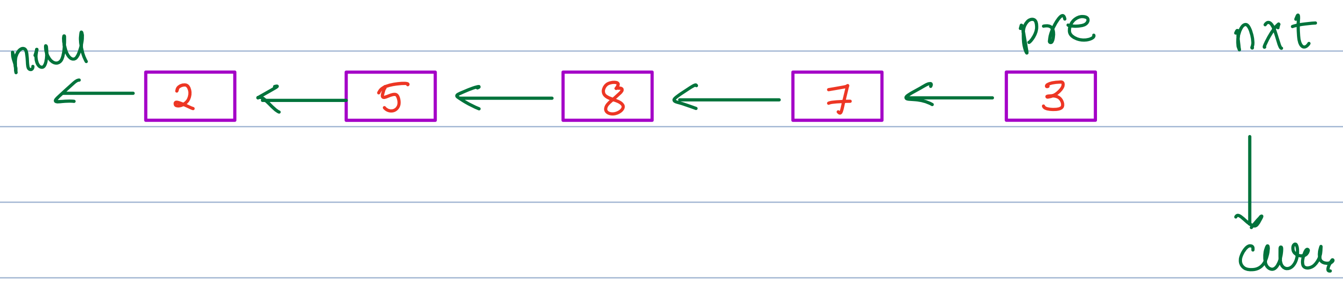


Bruteforce : Use extra space

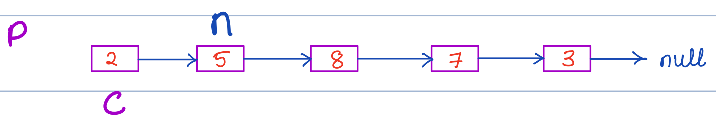
→ Store linked list into an array

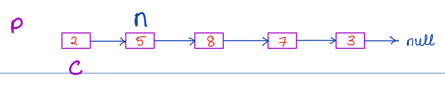
→ Reverse the array

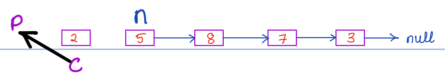
→ Create a new linked list.

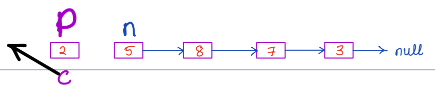


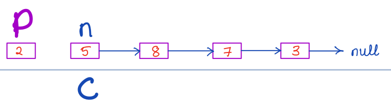
```
Node  reverseList (Node head ) {
        if (head == null)  return head

        pre = null
        cur = head

        while ( cur != null ) {
            nxt = cur.next

            cur.next = pre

            pre = cur

            cur = nxt
        }

        return pre
}
```











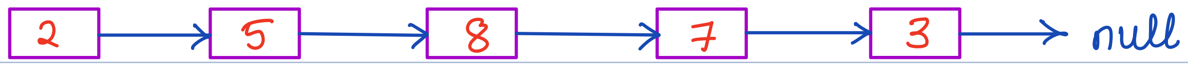
TC : O(N)

SC : O(1)

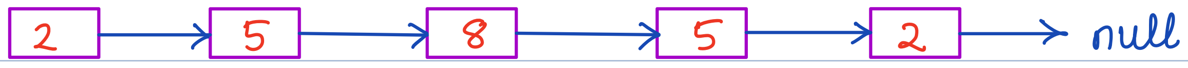
Q ⟶ Check if the given linked list is a palindrome.

madam

mom



return False



return true

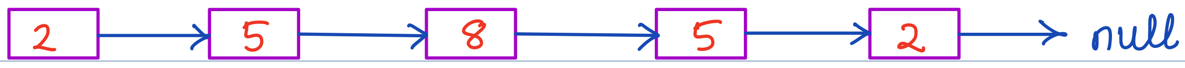
Idea 1 ⟶ Create a new linked list copy

Reverse the copy

compare copy with original linked list.

a b b a

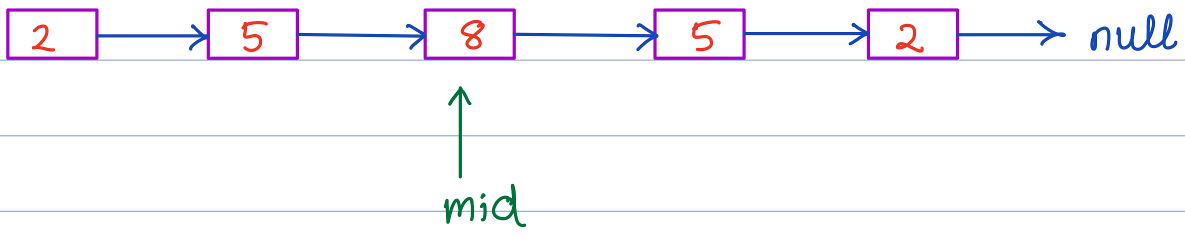
a b



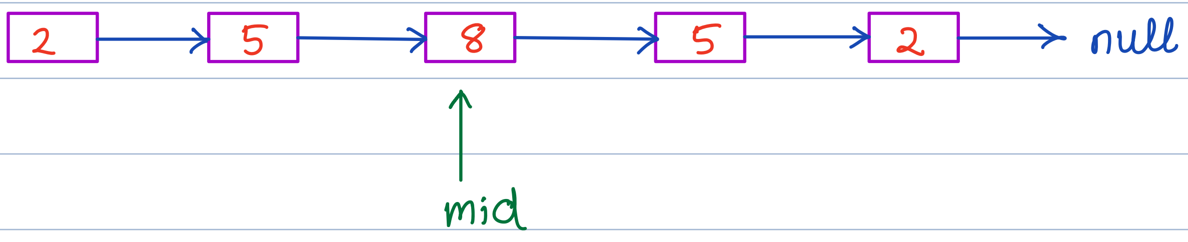
Step 1 ⟶ Find middle of linked list

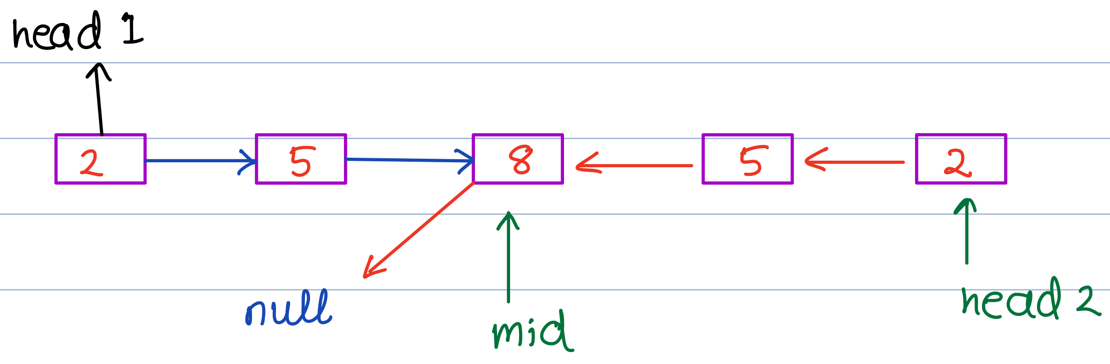
First count no. of nodes = size

Iterate to the node size/2 .



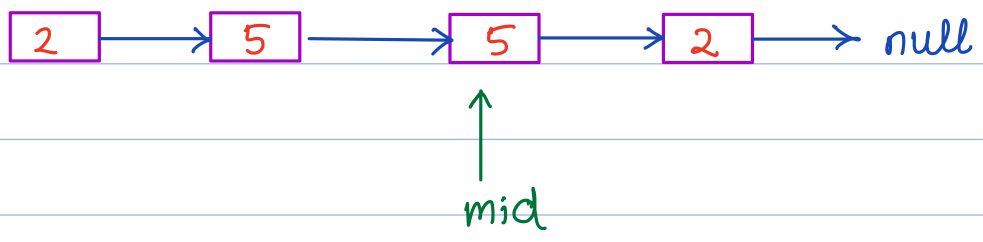
step 2 ⟶ Reverse the linked list from mid using previous code.

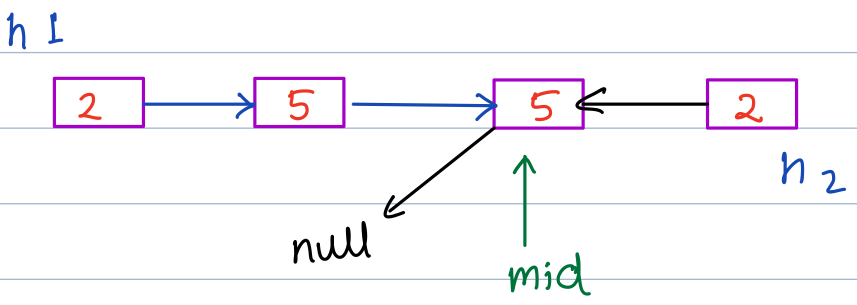




step 3 ⟶ Check head 1 and head 2 values
if they are not same return false
else

head1 = head1.next
head2 = head2.next.

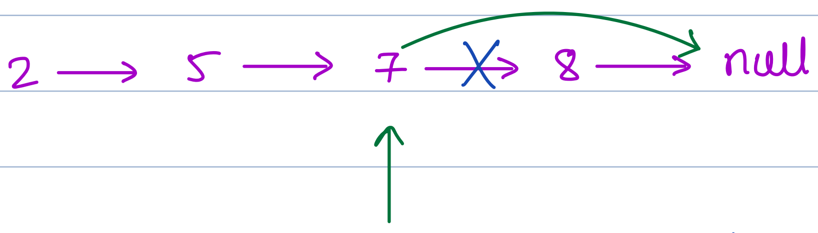




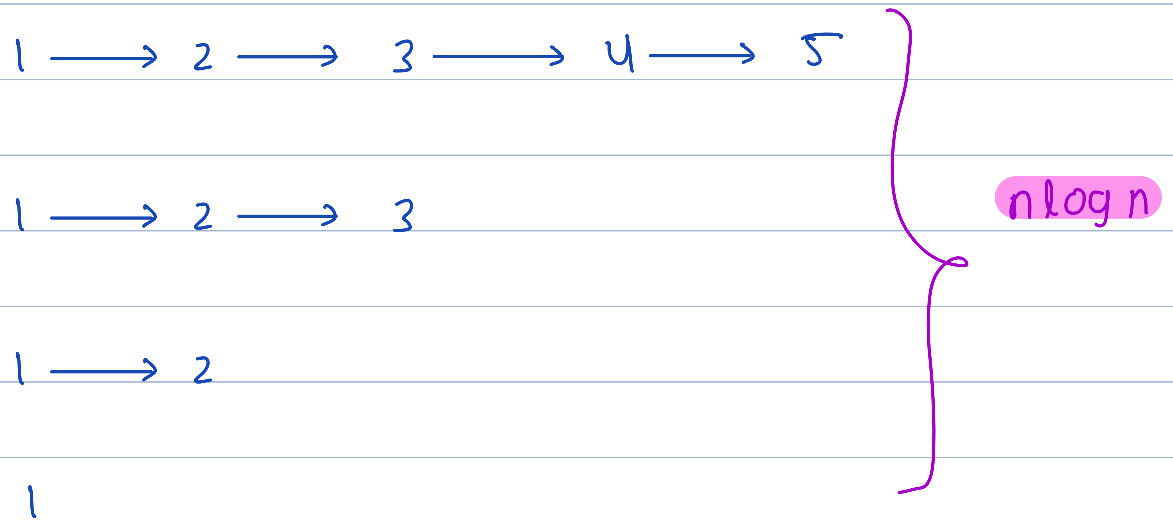
TC : O(N)
SC : O(1)

TODO ⟶ Write the code.

Doubt session



```
temp = head

while ( temp.next != null) {
    if ( temp.next. val == x ) {
        temp.next = temp.next. next
        return head
    }
    temp = temp.next
}
```


1 → 2 → 3 → 4 → 5
1 → 2 → 3
1 → 2
1
nlog n