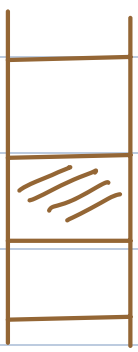
SCALER

LinkedList - 1

TABLE OF CONTENTS

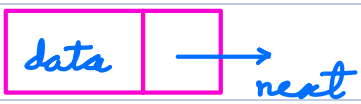
Issues with Array

Continuous memory allocation.

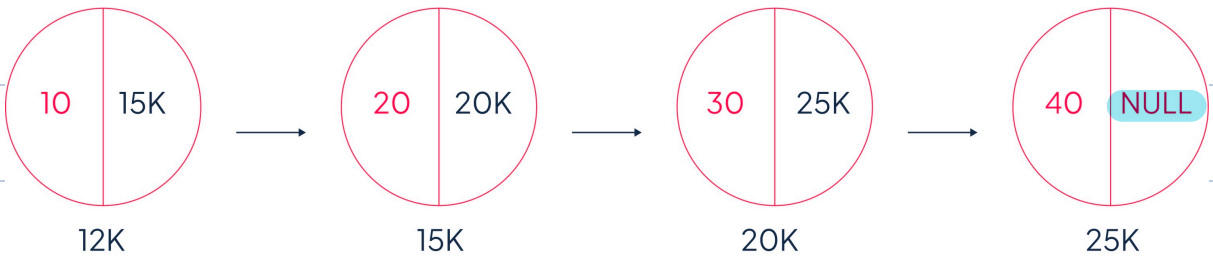


Linked List → Linear Data Structure

Non-continuous memory allocation.



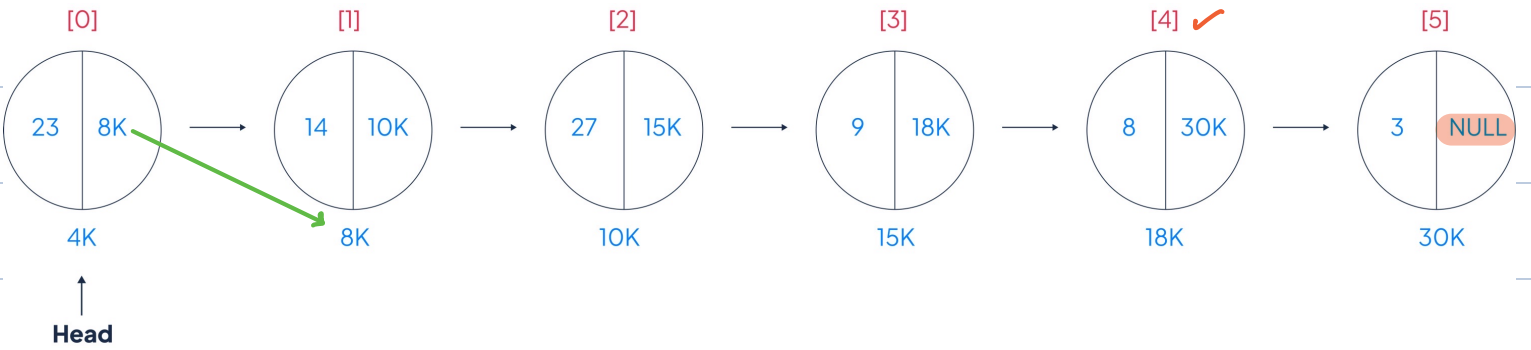
LinkedList



Head

```
class Node {
        int val;
        Node next;
        public Node(int v);
                this.val = v;     next = null;
        }
}
```

How to access the element at kth.idx in a linked-list

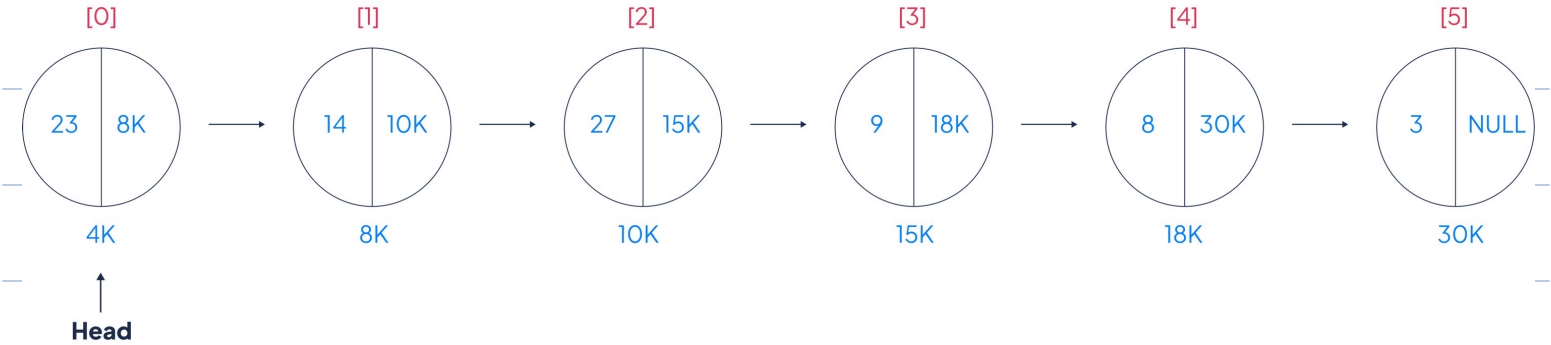


idx = 4 → 8

```
if (Head == null)    return -1
temp = Head
for i → 1 to K {
    temp = temp.next
    if (temp == null)    return -1
}
return temp.val
```

TC = O(K)

Search in linked-list



search(3) → true

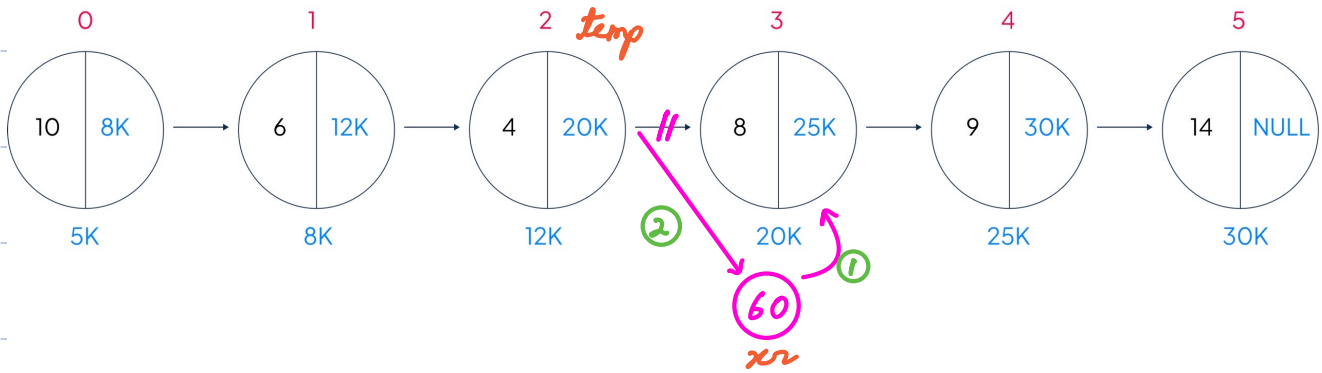
X

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

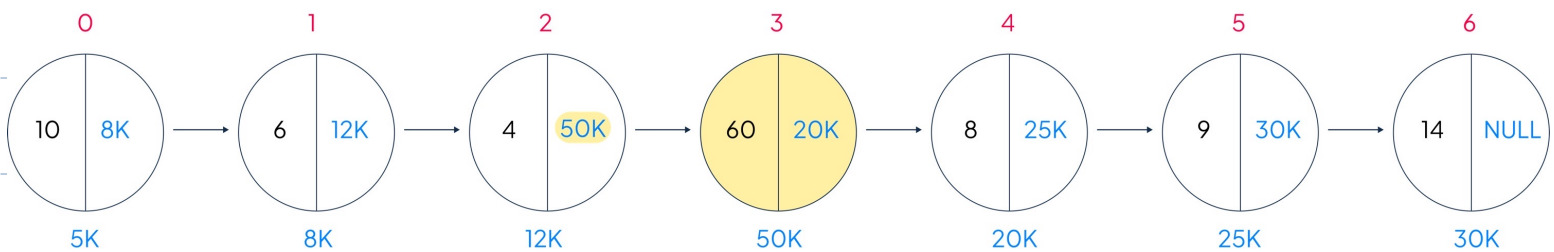
TC = O(N) // Linear Search

Binary Search → Not possible ∵ we cannot jump to middle element.

Insertion in linked-list at Kth index

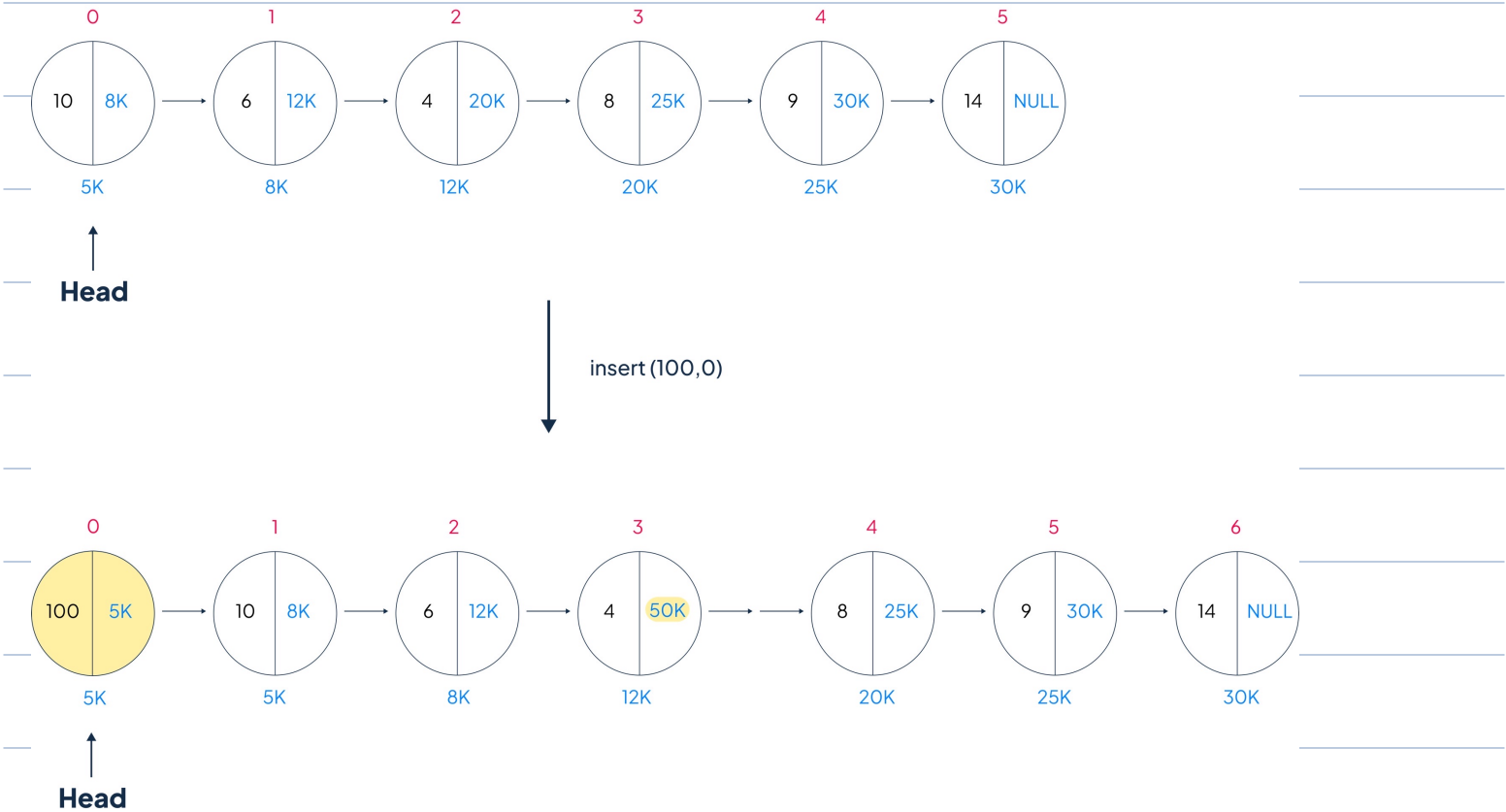


idx value
insert At (3,60)

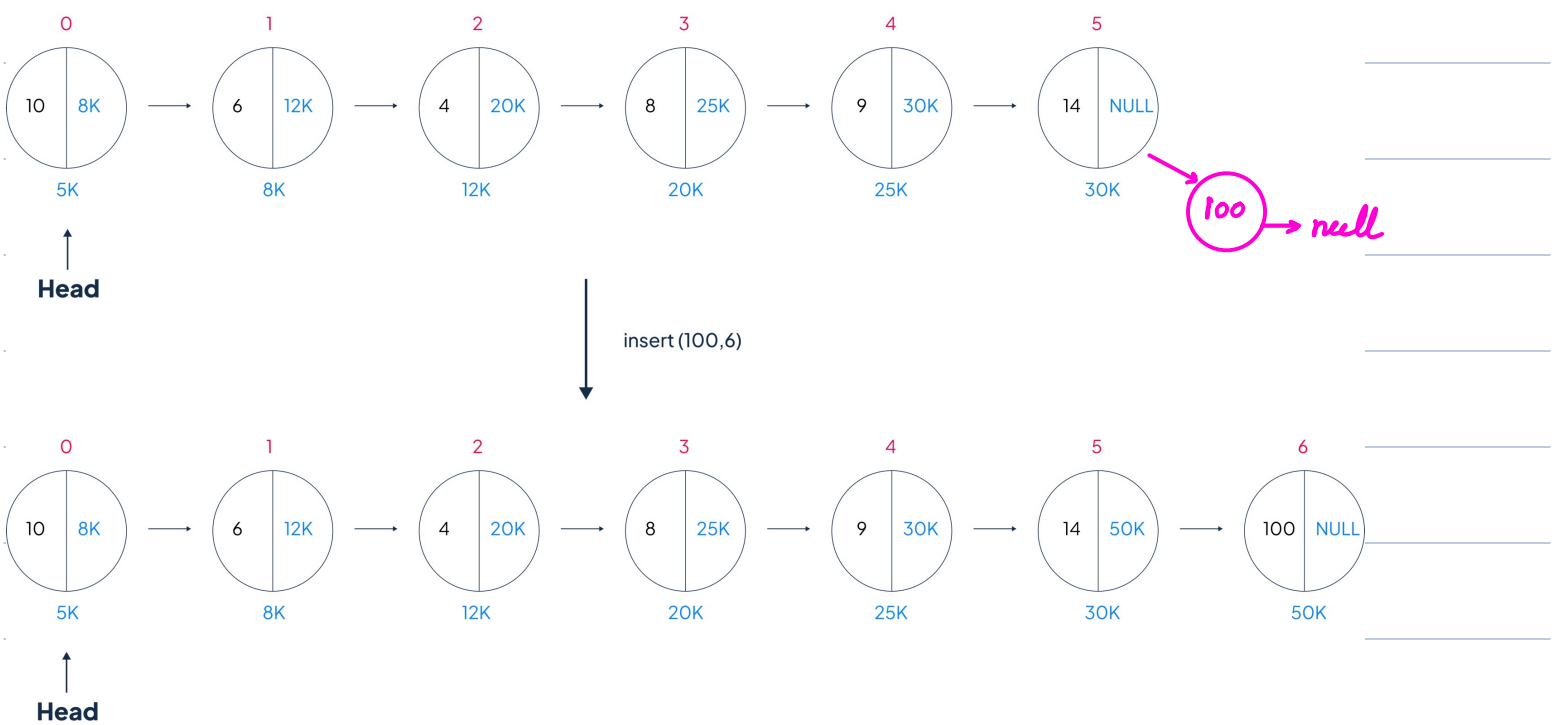


Edge-cases

If K = 0



If K = N



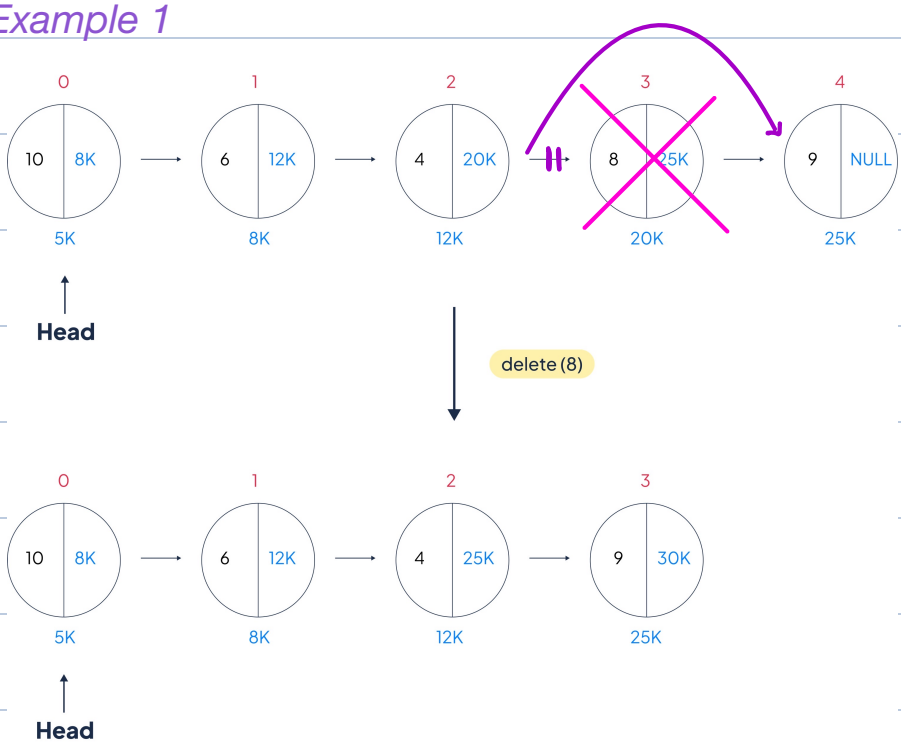
```
// insert (X, K)
Node xr = new Node (X)
if (K == 0) {
    xr.next = Head
    Head = xr
    return Head
}
temp = Head
for i → 1 to (K-1) {
    temp = temp.next
}
xr.next = temp.next
temp.next = xr
return Head
```

TC = O(K)

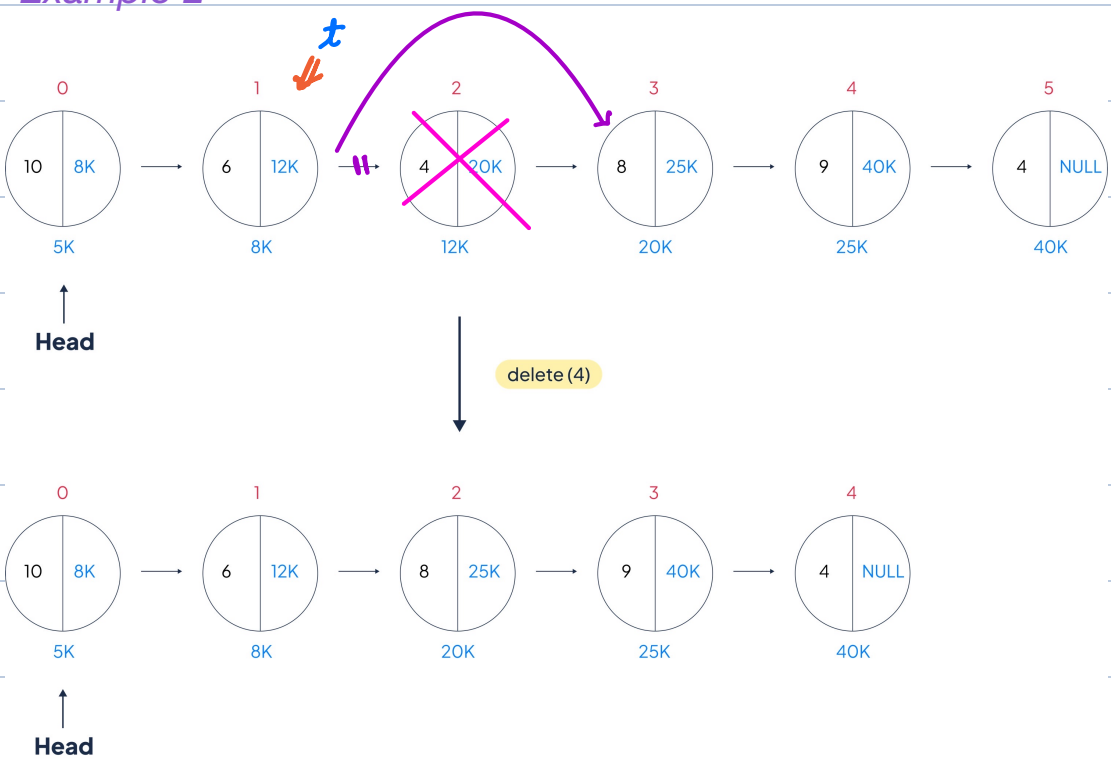
Deletion in linked-list

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

Example 1



Example 2



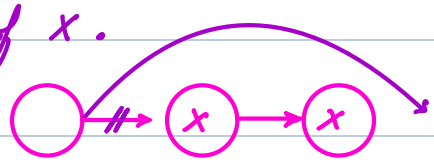
```
if ( Head == null)  return null
if ( Head.val == x ) { Head = Head.next
                       return Head }

temp = Head
while ( temp.next != null ) {
    if ( temp.next.val == x ) {
        temp.next = temp.next.next
        break
    }
    temp = temp.next
}

return Head
```

TC = O(N)

H.W → Delete all occurrences of x.



Scenario

Comment

OnePlus has a lineup of N mobile phones ready in their manufacturing line. It has detected a defect in one of their phone models during production. They have decided to recall all phones of the defective model from their manufacturing line. Your task is to help OnePlus remove all defective phones from their production lineup efficiently

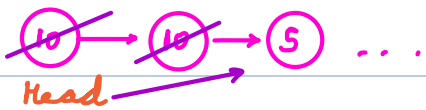
Problem

You are given a linked list A of N nodes where each node represents a specific model type of a OnePlus mobile phone in the manufacturing line. Each node contains an integer representing the model number of the phone. You will also be given an integer B which represents the model number of the defective phone that needs to be removed.

Your goal is to remove all nodes (phones) from the linked list that have the model number B and return the modified linked list representing the updated manufacturing line.

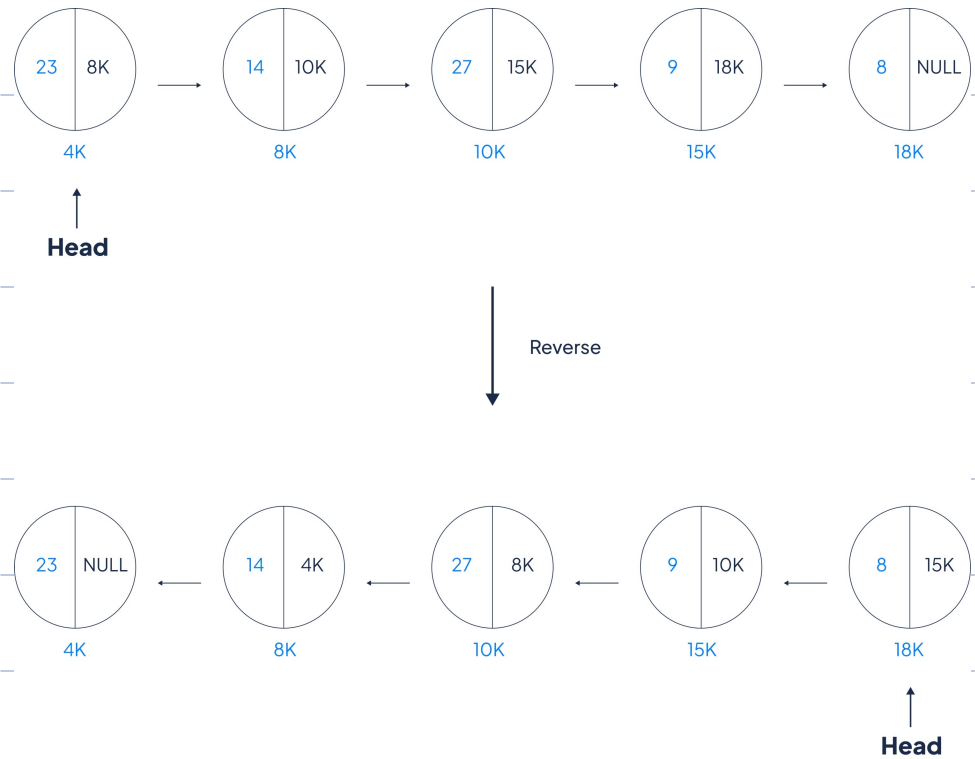
Delete all occurrences of B from list A.

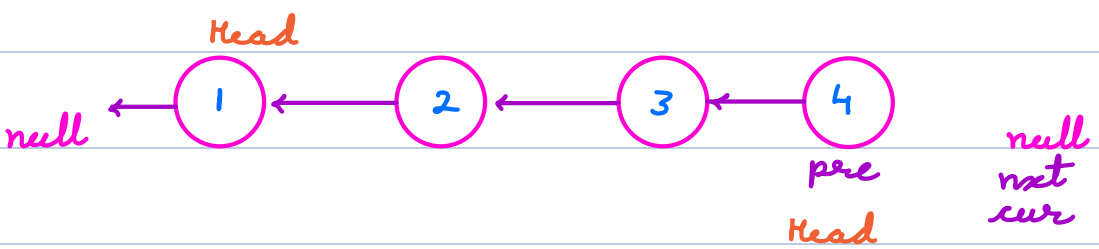
```
if ( Head == null) return null
while ( Head != null && Head.val == B )
        Head = Head.next          // B = 10
temp = Head
while ( temp.next != null ) {
    if ( temp.next.val == B )
        temp.next = temp.next.next
    else
        temp = temp.next
}
return Head
```



TC = O(N)

Reverse a linked-list



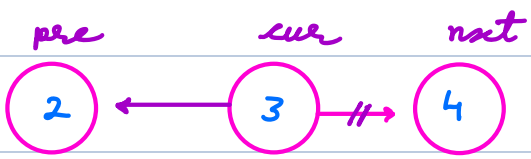


nxt = cur.next

cur.next = pre ✓

pre = cur

cur = nxt



```
cur = Head
pre = null
while (cur != null) {
    nxt = cur.next
    cur.next = pre
    pre = cur
    cur = nxt
}
return pre  // head
```

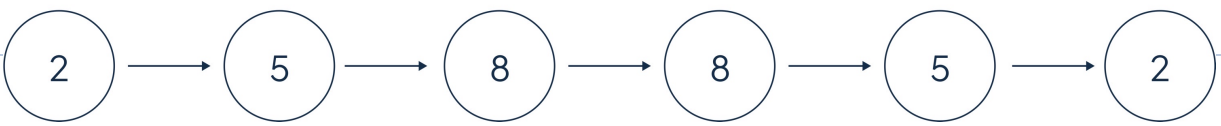
TC = O(N) SC = O(1)

- Check if a linked-list is a palindrome or not

"racecar" → = ←

X

✓



✓

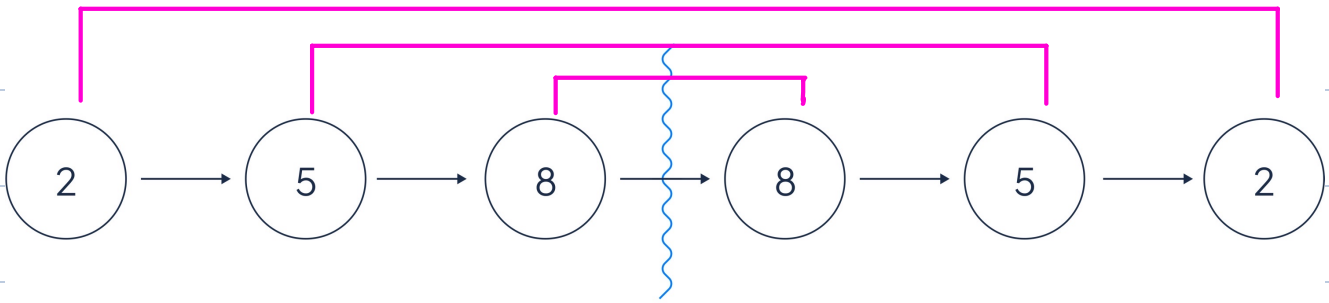
✓

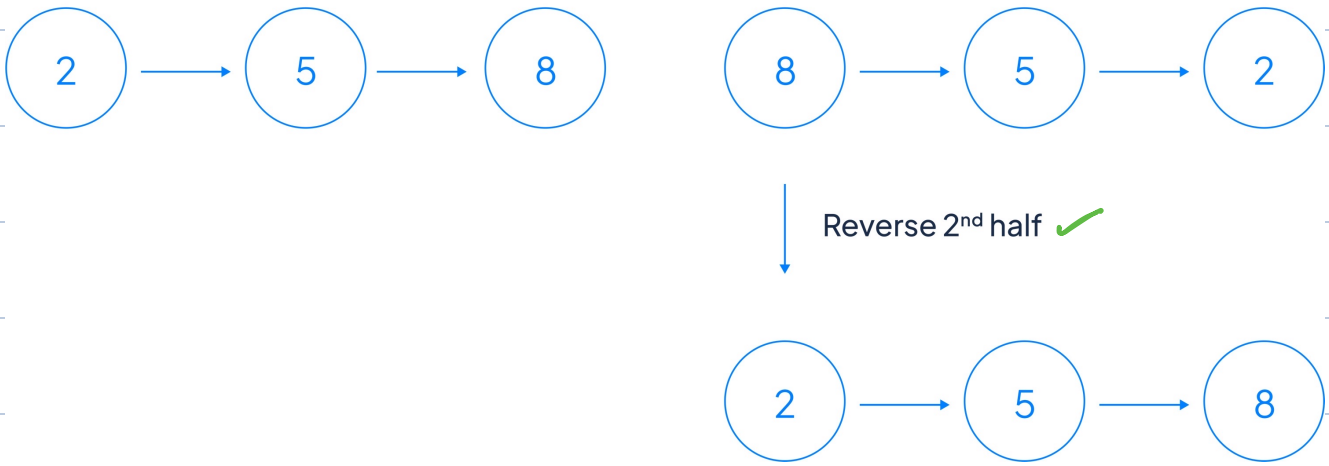
Sol 1 → 1) create copy of list

2) Reverse the copy

3) check if reverse of copy & original list is same.

TC = O(N) SC = O(N) → copy





1) Find middle element & divide the list. → Find size & travel half. ✓

2) Reverse second half. ✓ TC = O(N)

3) compare first half with reversed second half. ✓

TC = O(N) SC = O(1)

If first half and reversed second half are equal

→ return true;

otherwise

→ return false;