CODING SHUTTLE

Week 9 LIVE

Advanced LinkedList Problems And Doubts Session

In This Lecture

1. Partition List
2. Longest Palindrome List

Partition List

Given a linked list A and a value B, partition it such that all nodes less than B come before nodes greater than or equal to B.

You should preserve the original relative order of the nodes in each of the two partitions.

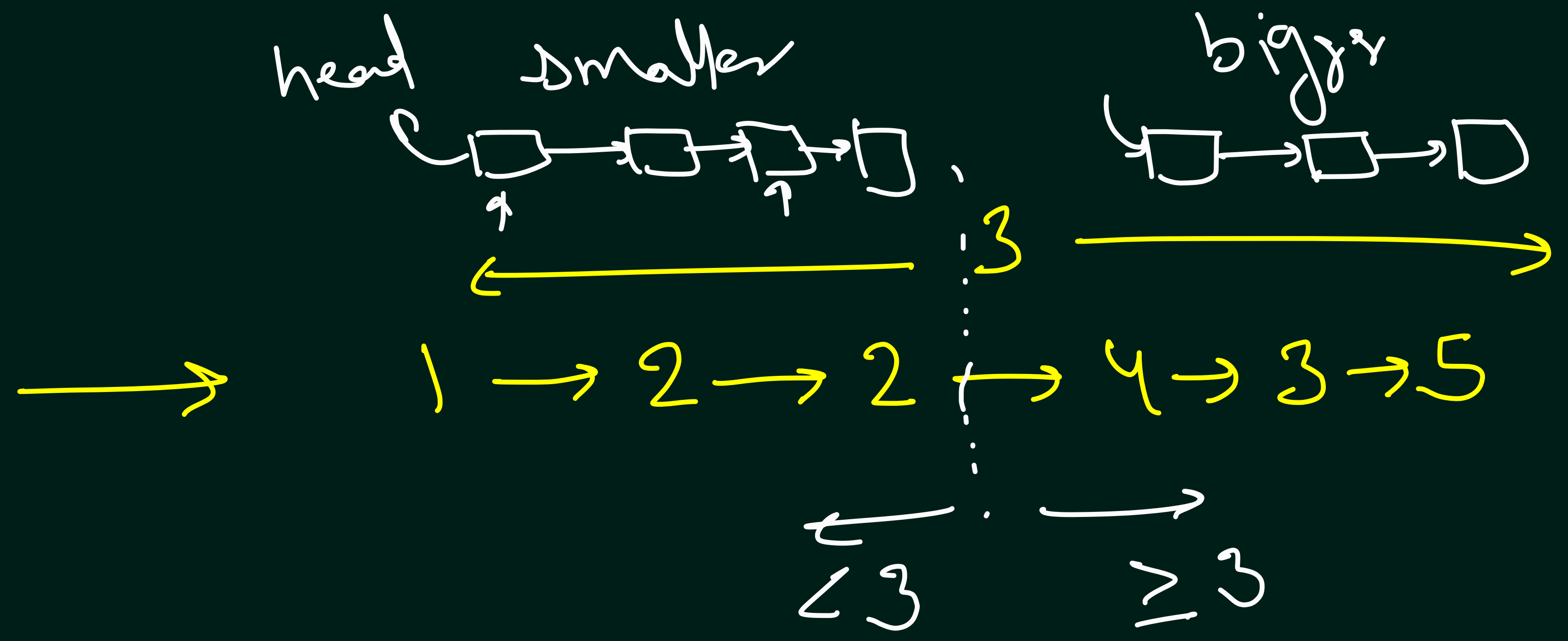
Input:

A = [1, 4, 3, 2, 5, 2]

B = 3

Output:

[1, 2, 2, 4, 3, 5]



Partition List

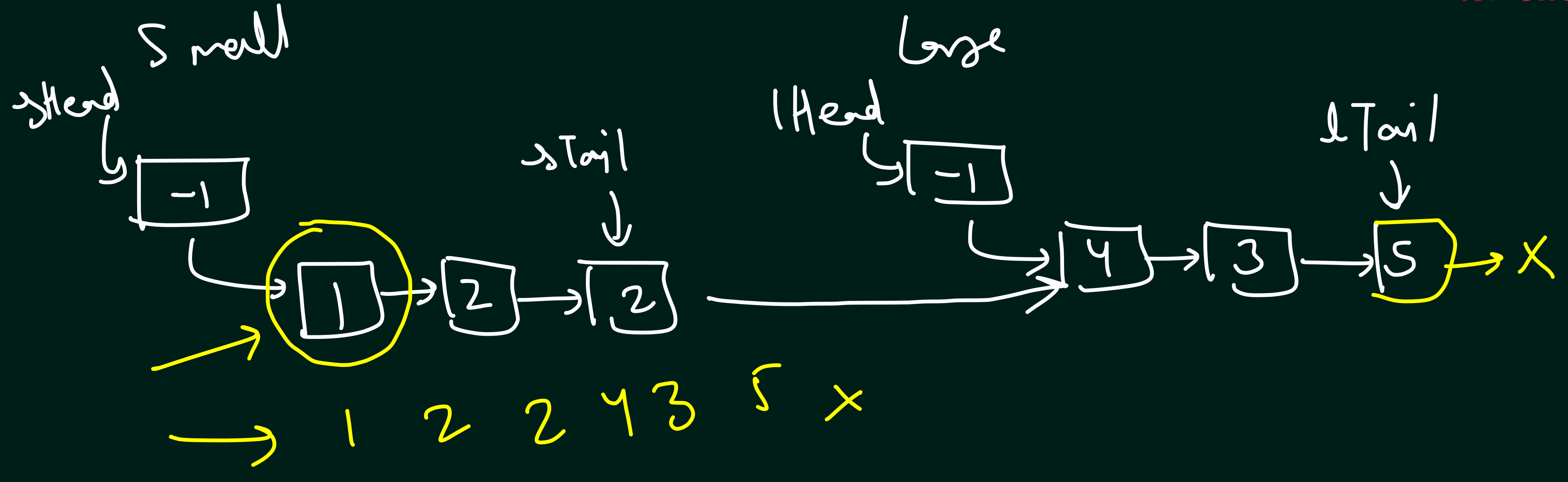
Input:

A = [1, 4, 3, 2, 5, 2]

B = 3

Output:

[1, 2, 2, 4, 3, 5]



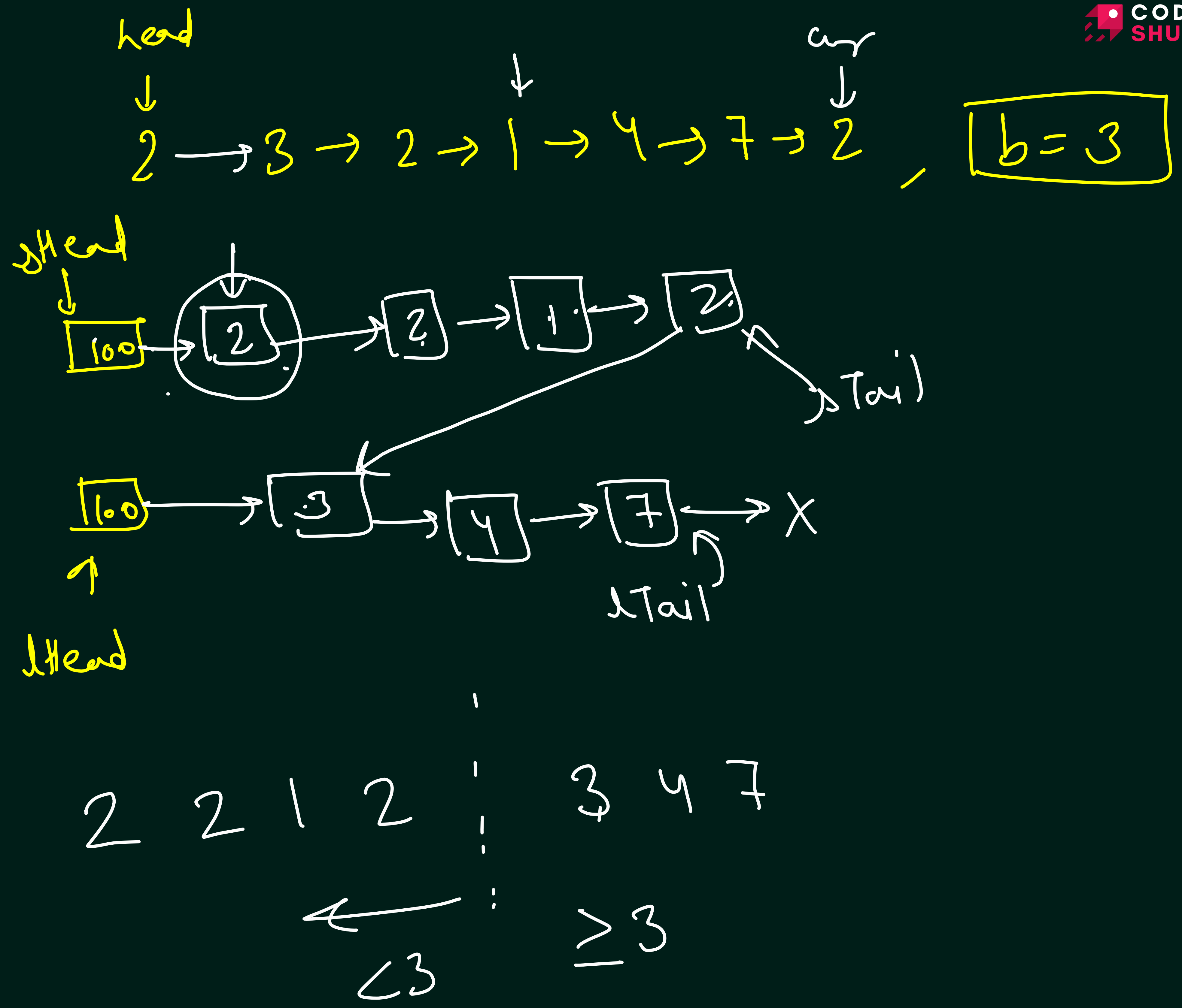
sTail. next = lHead. next ;

Return?

sHead. next

Partition List

```java
static Node partitionList(Node head, int b) {
    if(head == null || head.next == null) return head;
    Node sHead = new Node( data: -1);
    Node lHead = new Node( data: -1);

    Node sTail = sHead;
    Node lTail = lHead;
    Node cur = head;
    while (cur != null) {
        if(cur.data < b) {
            sTail.next = cur;
            sTail = cur;
        } else {
            lTail.next = cur;
            lTail = cur;
        }
        cur = cur.next;
    }

    sTail.next = lHead.next;
    lTail.next = null;
    return sHead.next;
}
```



Longest Palindrome List

Given a linked list , return the length of the longest palindrome list that is present in the given linked list.

Input:

head = [1 -> 2 -> 3 -> 3 -> 2 -> 4]

Output:

4

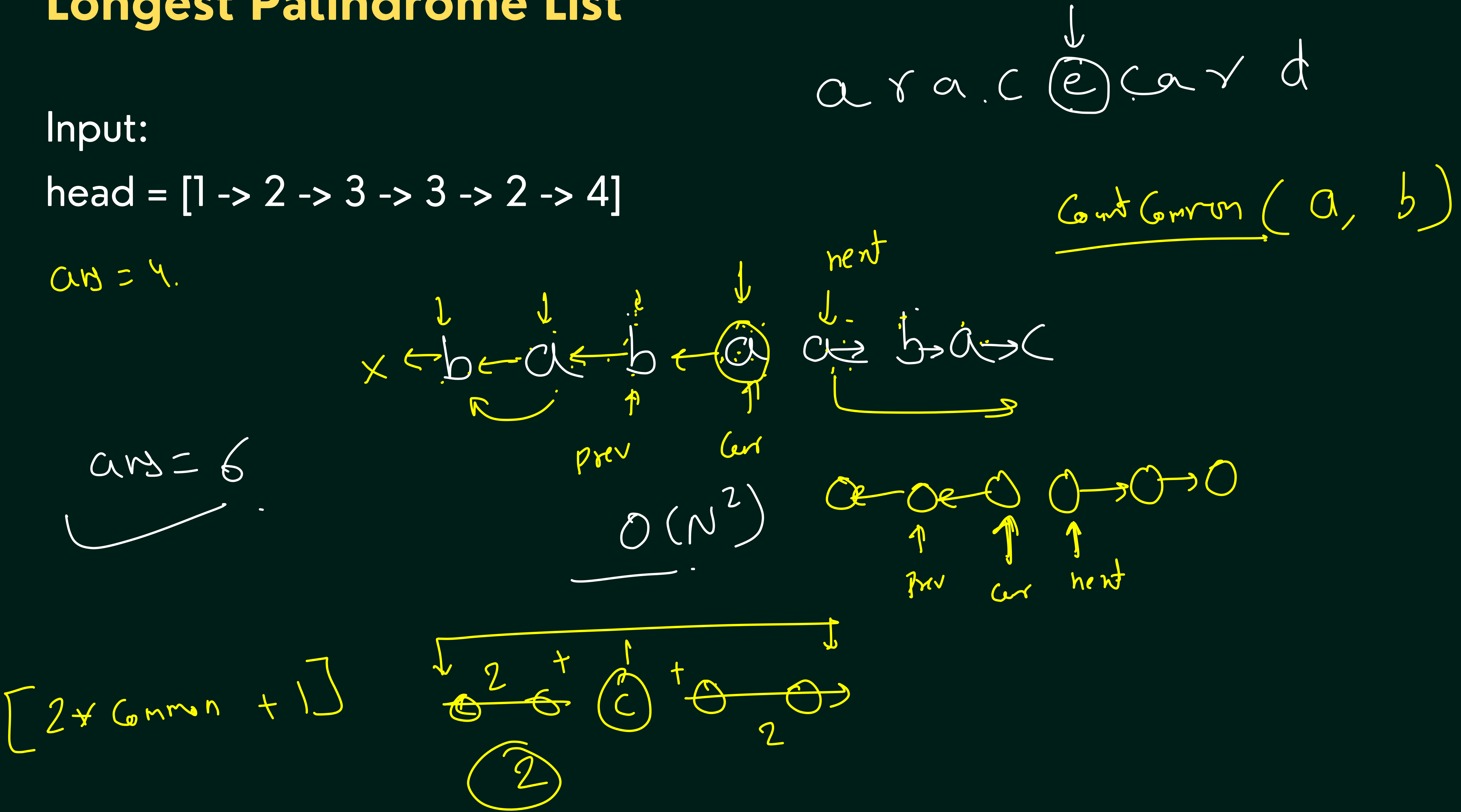
Explanation:

2 -> 3 -> 3 -> 2 is the length of the longest palindrome in the list.

Longest Palindrome List

Input:

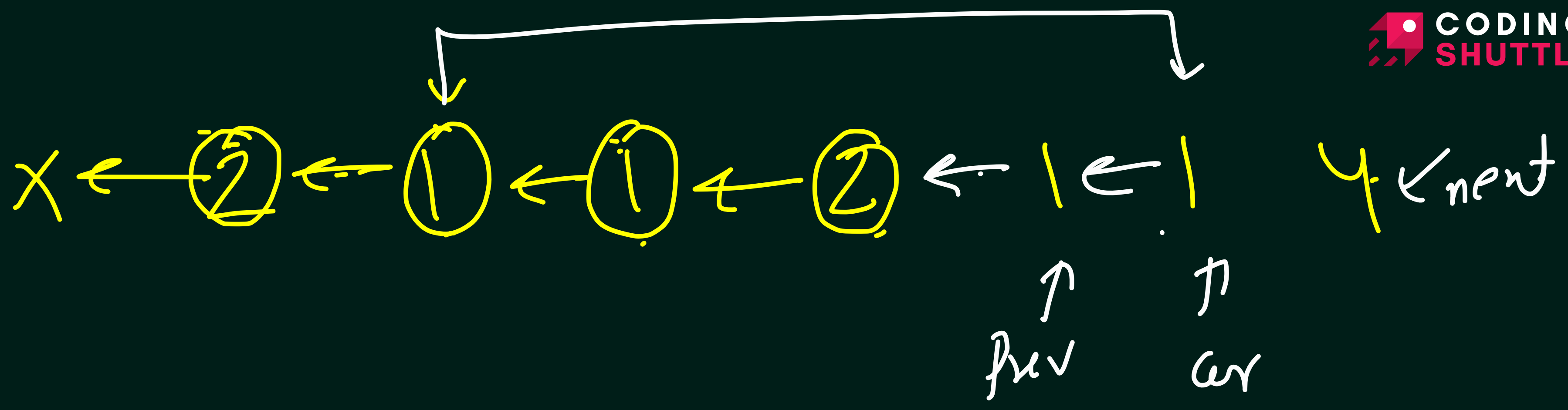
head = [1 -> 2 -> 3 -> 3 -> 2 -> 4]



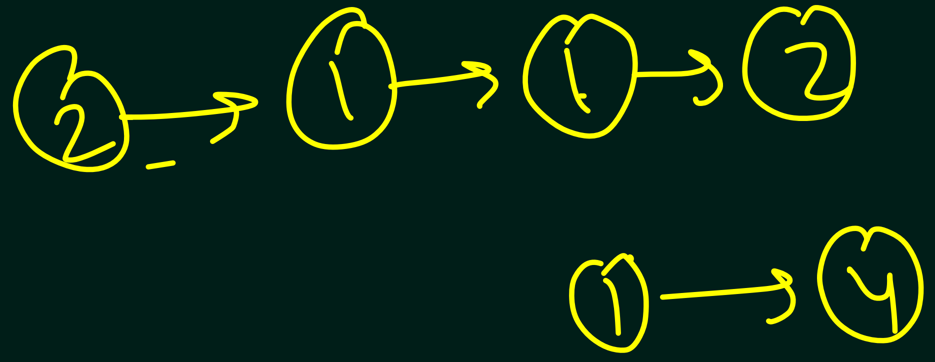
Longest Palindrome List

CODING SHUTTLE

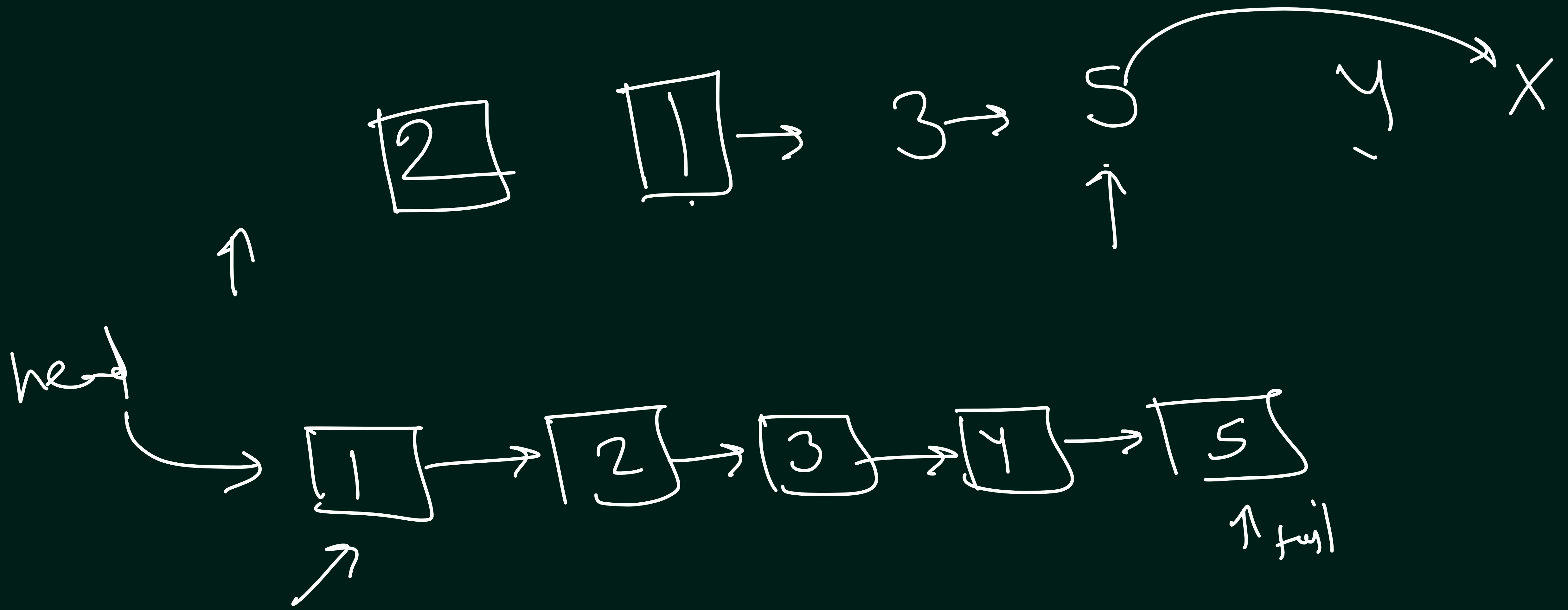
```java
static int longestPalindrome(Node head) {
    if(head == null) return 0;
    if(head.next == null) return 1;

    Node cur = head;
    Node prev = null;
    int ans = 0;

    Node ansNode = null;


    while (cur != null) {
        Node next = cur.next;
        cur.next = prev;
        int commonIfCurIsExactCenter = countCommon(prev, next);
        int lengthFromExactCenter = 2 * commonIfCurIsExactCenter + 1;

        int commonIfCurIsNotExactCenter = countCommon(cur, next);
        int lengthFromNotExactCenter = 2 * commonIfCurIsNotExactCenter;

        int largerOfTheseTwoLengths = Math.max(lengthFromExactCenter,
                lengthFromNotExactCenter);
        ans = Math.max(ans, largerOfTheseTwoLengths);
        prev = cur;
        cur = next;
    }
    return ans;
}
```



ans = ~~4~~ 5



Q. Sort a Linked List


2
1
3
5
4
X
head
1
2
3
4
5
tail