Brush Size

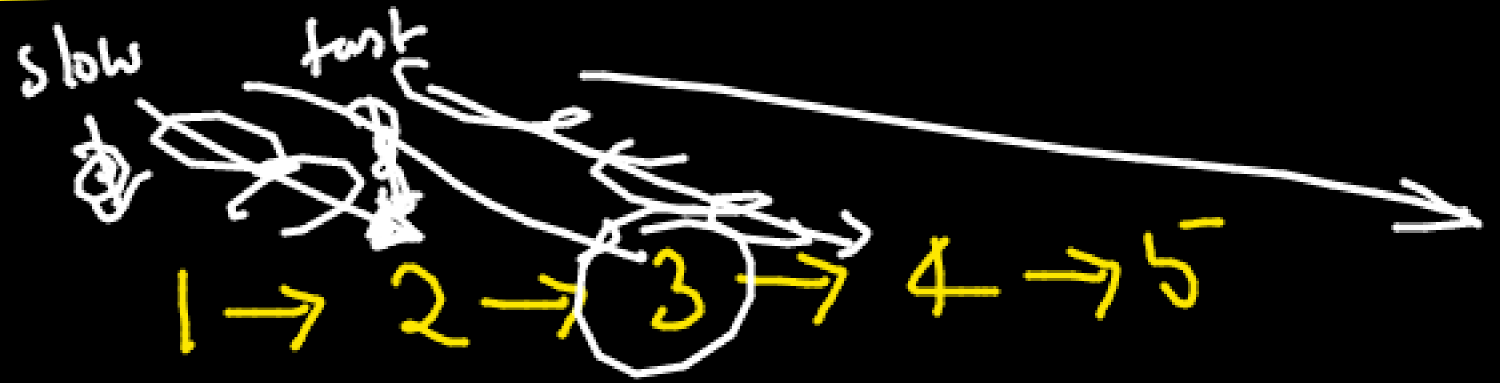
Midpoint of LL

10 → 20 → 30 → 40

⇒ ans = 20

10 → 20 → 30 → 40 → 50

⇒ ans = 30

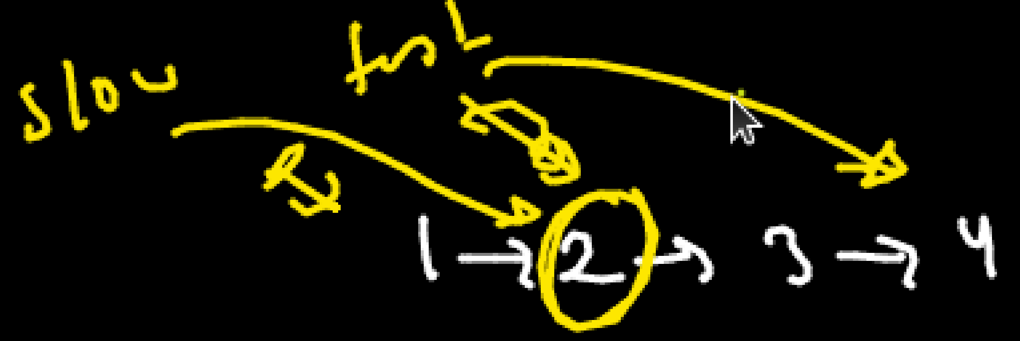


fast != NULL &&
fast->next != NULL

1 → 2 → 3 → 4 → 5

code ↓

```
Node * slow = head;
      *fast = head->next;
```



Brush Size

Code of midpoint

```
Node* midpoint (Node *head) {
    Node *slow = head;
    Node *fast = head->next;
    while (fast != NULL && fast->next != NULL) {
        fast = fast->next->next;
        slow = slow->next;
    }
    return slow;
}
```

$1 \to 2 \to \boxed{3} \to 4 \to 5 \to \text{null}$ odd

$1 \to \boxed{2} \to 3 \to 4 \to \text{null}$ even

Important Question

(i) Midpoint of LL

(ii) merge two sorted LL

(iii) Merge sort LL

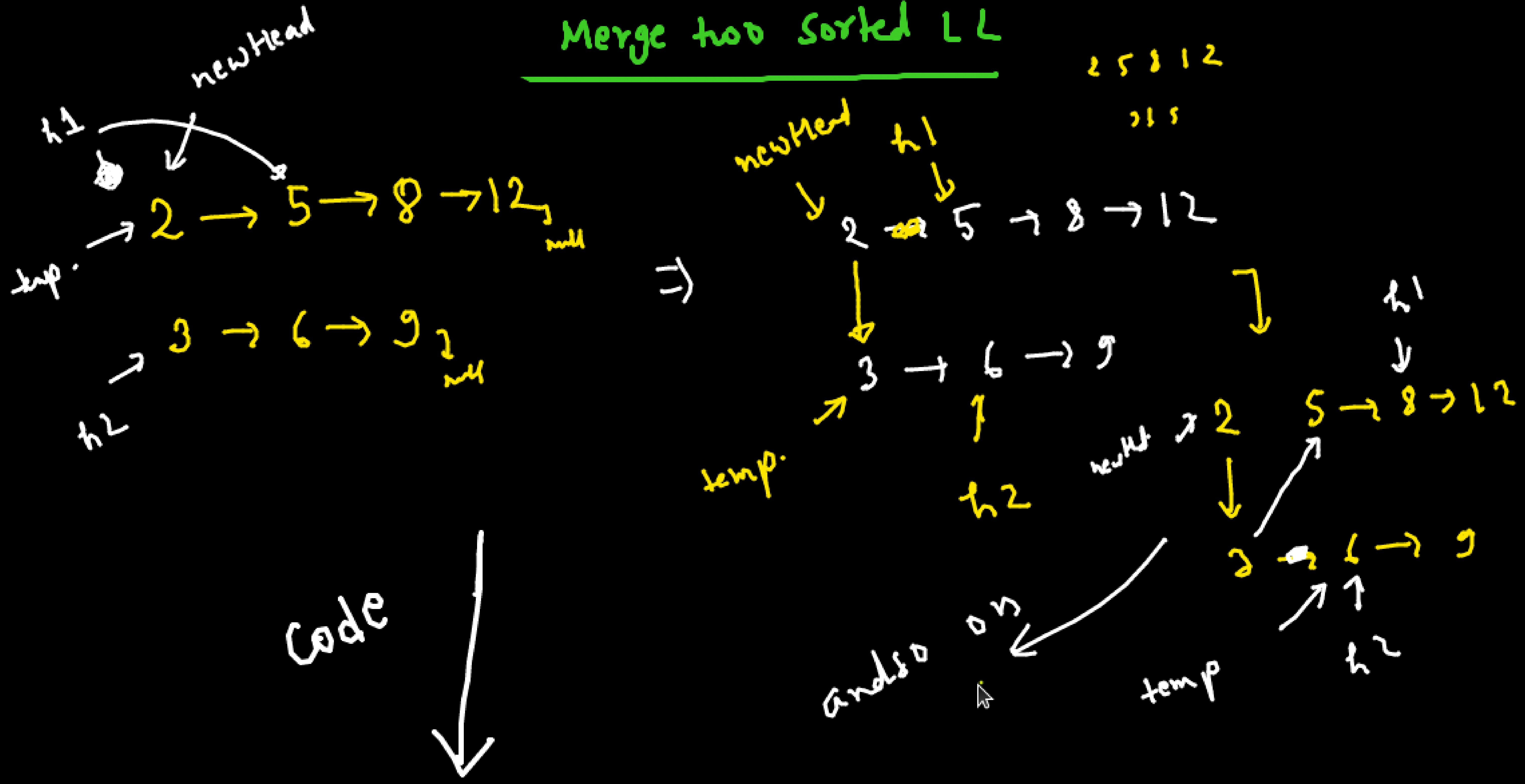
(iv) Reverse LL (recursive)

(v) delete every N Nodes

(vi) K Reverse

(vii) swap two Node

Merge two Sorted LL



Brush Size

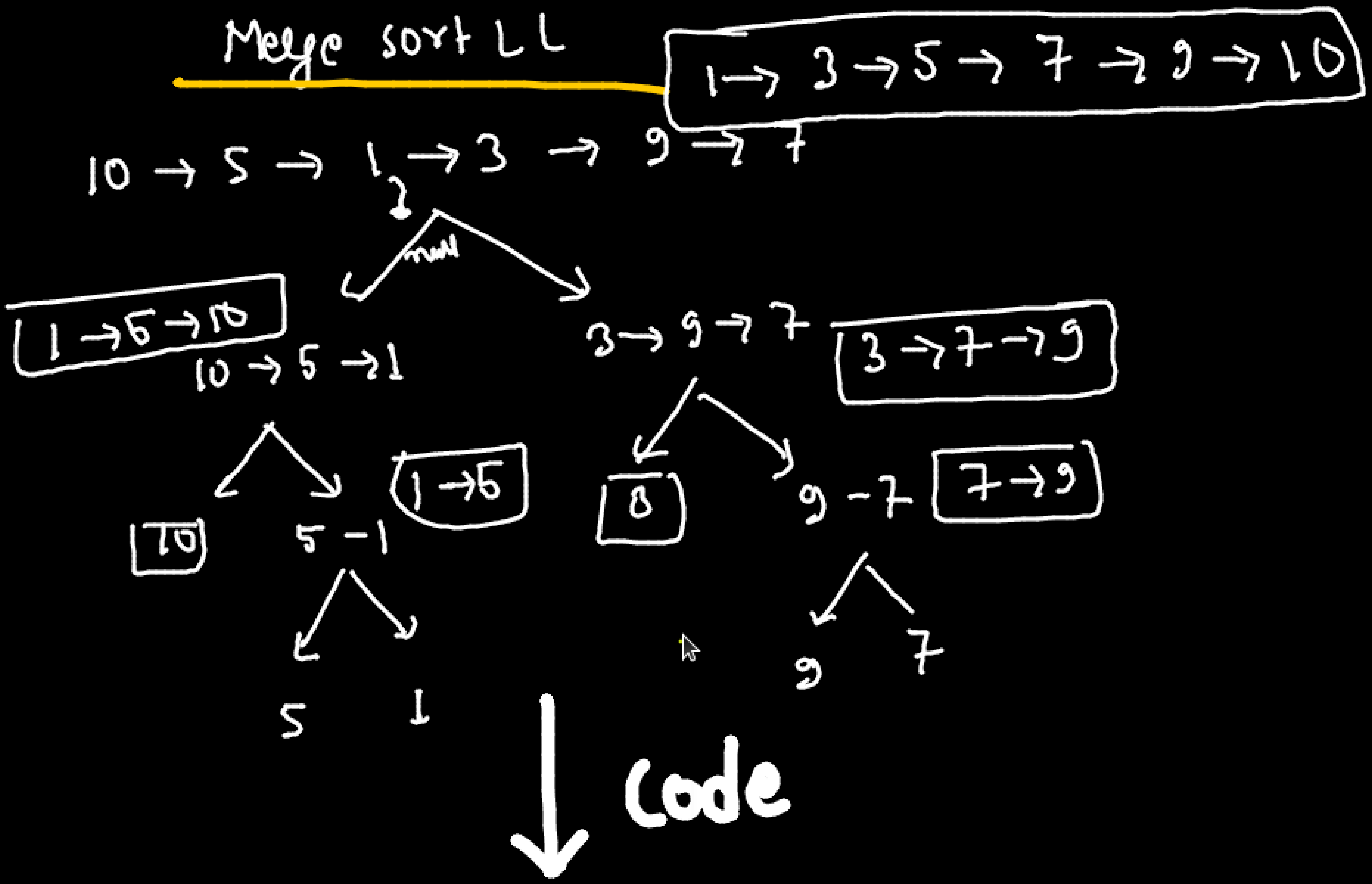
Code of Merge two sorted LL

(same as merge two sorted arrays)

newHead ko find kro and head1 ya head2 ko next karo

```
while (head1 != NULL an head2 != NULL)
{
        head1->data < head2->data
        {

        }
      an
        {

        }
}
```

```
while (head1 != NULL)
{

}

while (head2 != null)
{

}

return newHead;
```


Merge sort LL
1 → 3 → 5 → 7 → 9 → 10
10 → 5 → 1 → 3 → 9 → 7
null
1 → 5 → 10
10 → 5 → 1
3 → 9 → 7
3 → 7 → 9
1 → 5
10
5 - 1
5
1
3
9 - 7
7 → 9
9
7
code

Brush Size

```
Node * mergeSort (Node *head){
    if (head == NULL || head->next == NULL)
            return head;

    Node *mid = middle (head);

    Node * first = head;
    Node * second = mid->next;
      mid->next = NULL;

    first = merge Sort (first )
    second = mergesort ( second )
        head = mergetwosortedLL (first, second)
}
```

$1 \to 2 \to \textcircled{3} \to 4 \to 5$

$1 \to \textcircled{2} \to 3 \to 4$

gives middle Node

becaus mid element have to be NULL.

merge two sorted LL

return head;

Brush Size

Solution - I $O(N^2)$

Reverse LL (Recursive) - 1

```
Node* reverse (Node *head) {
  if (head==NULL or head->next==NULL)
          return head;

  Node * smallAns = revese( head->next);

  Node *temp = smallAns;          }
  while (temp->next != NULL){      } Go to last node
        temp = temp->next;        }
  }

     temp -> next =head;
     head ->next =NULL;

  return smallAns;
}
```

head $\rightarrow$ 1 | $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ 5 $\rightarrow$ null

smallAns $\rightarrow$ 5 $\rightarrow$ 4 $\rightarrow$ 3 $\rightarrow$ 2 $\rightarrow$ null (temp at 2)

1 $\rightarrow$ null

Traverse and go to last

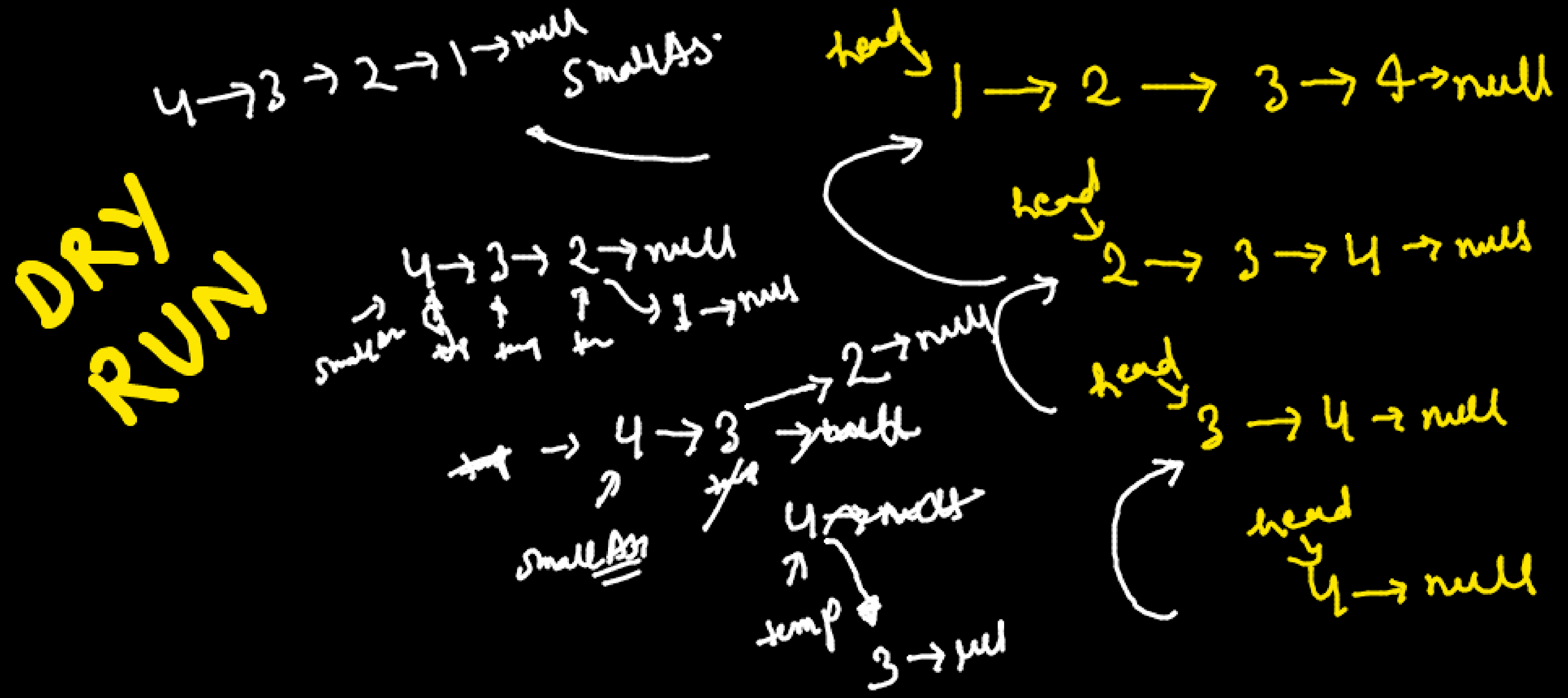
temp ->next=head

head ->next=NULL

$\downarrow$ eg.

9 15

Brush Size



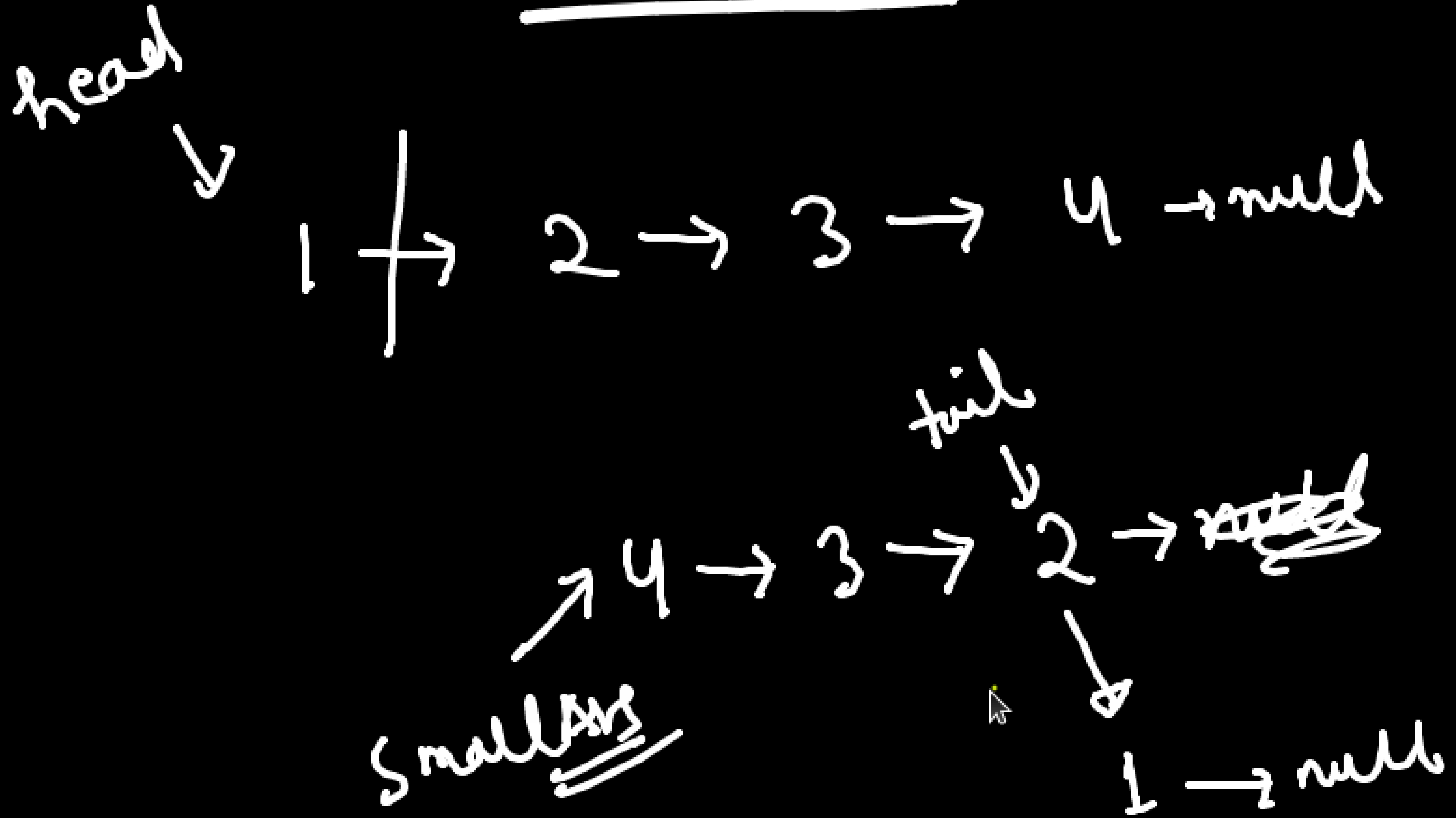
Reverse a LL (recursive) - 2 (Best)

$$\begin{tikzcd}
\text{head} \arrow[rd] & & & \\
& 1 \arrow[r, phantom, "|"] & 2 \arrow[r] & 3 \arrow[r] & 4
\end{tikzcd}$$

$$\begin{tikzcd}
& & & \text{tail} \arrow[dl] & \\
4 \arrow[r] & 3 \arrow[r] & 2 \arrow[rd, bend left] & & \\
\text{smallAns} \arrow[u] & & & 1 &
\end{tikzcd}$$

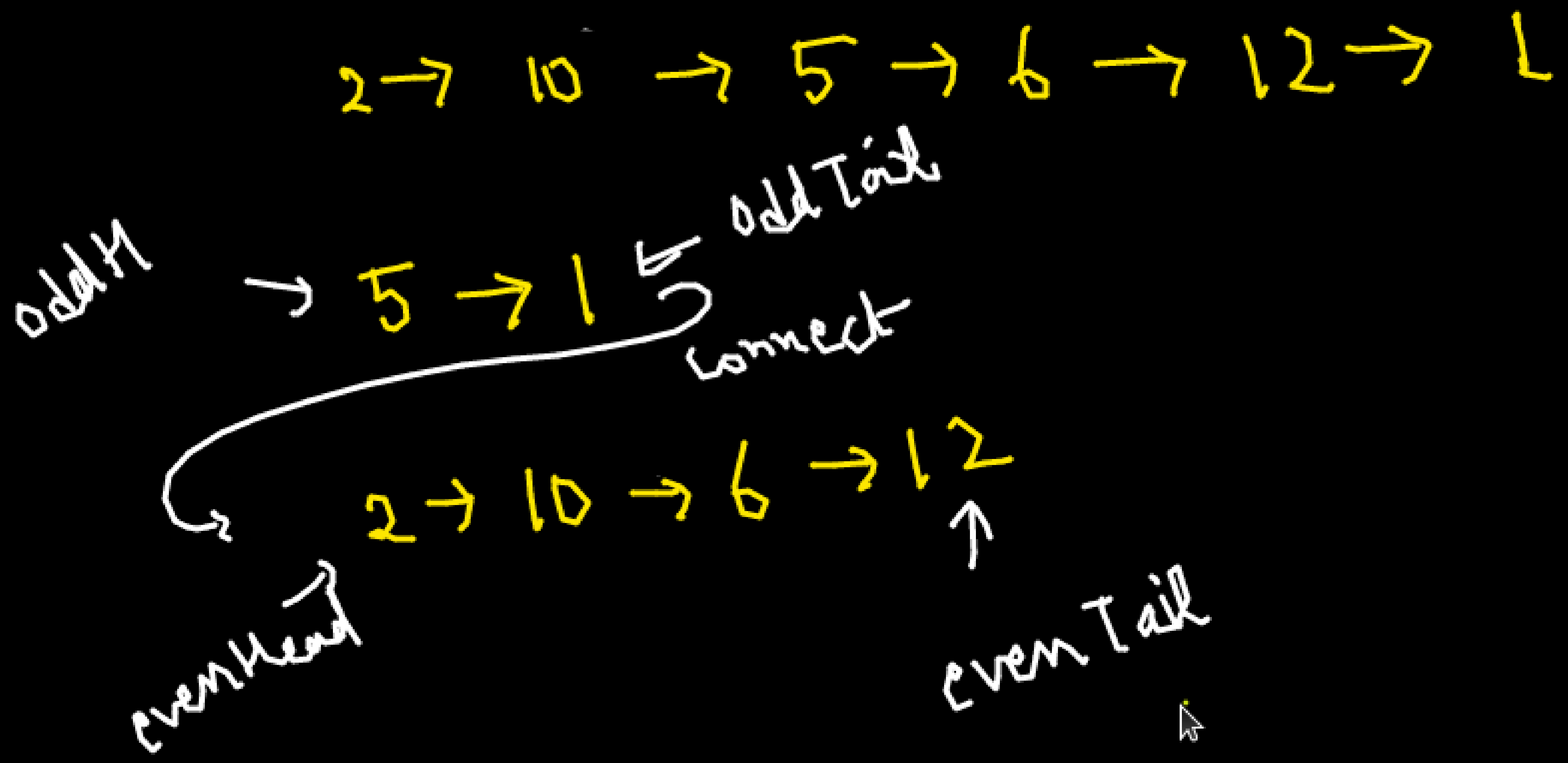
```
Node *reverse (Node *head) {
if (head==NULL or h->next==NULL)
    ret head

Node* smallAns = reverse(head->next);     smallAns

        {  Node *tail = head->next;
           tail->next = head;
           head->next = NULL
               return smallAns;

     my Rem ↓
}
```

DRY RUN



```
Node *tail = head->next
tail->next = head
head->next = NULL
return smallAns
```

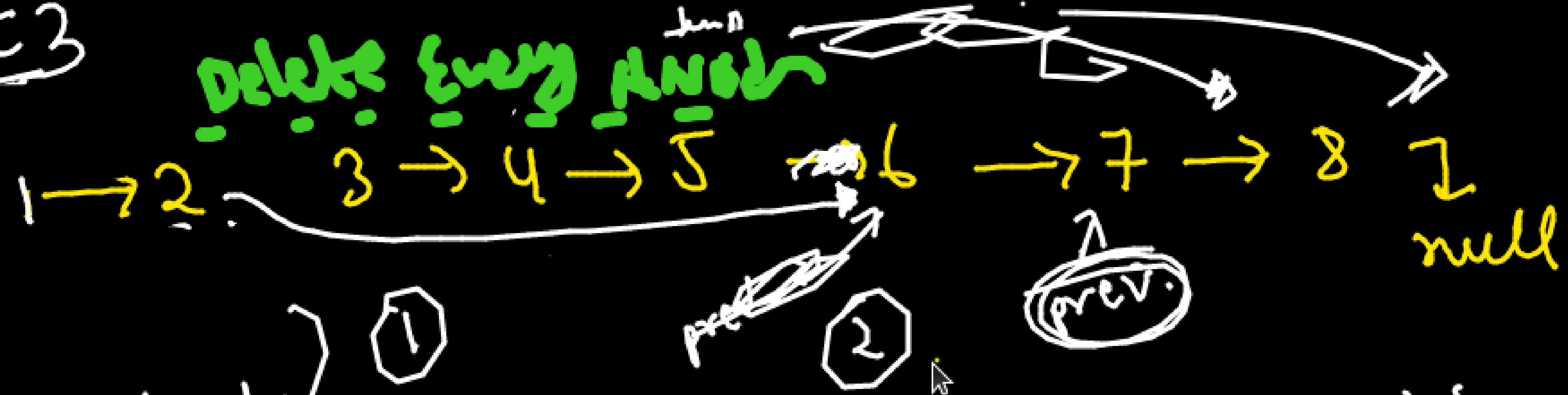
Even after odd LL



Brush Size

M=2 N=3

Delete Every Node



```
Node * temp = head;
Node * prev = head;

if (M==0) {
    return NULL;
}
```

1
0<2
1<2
2<5
3<5

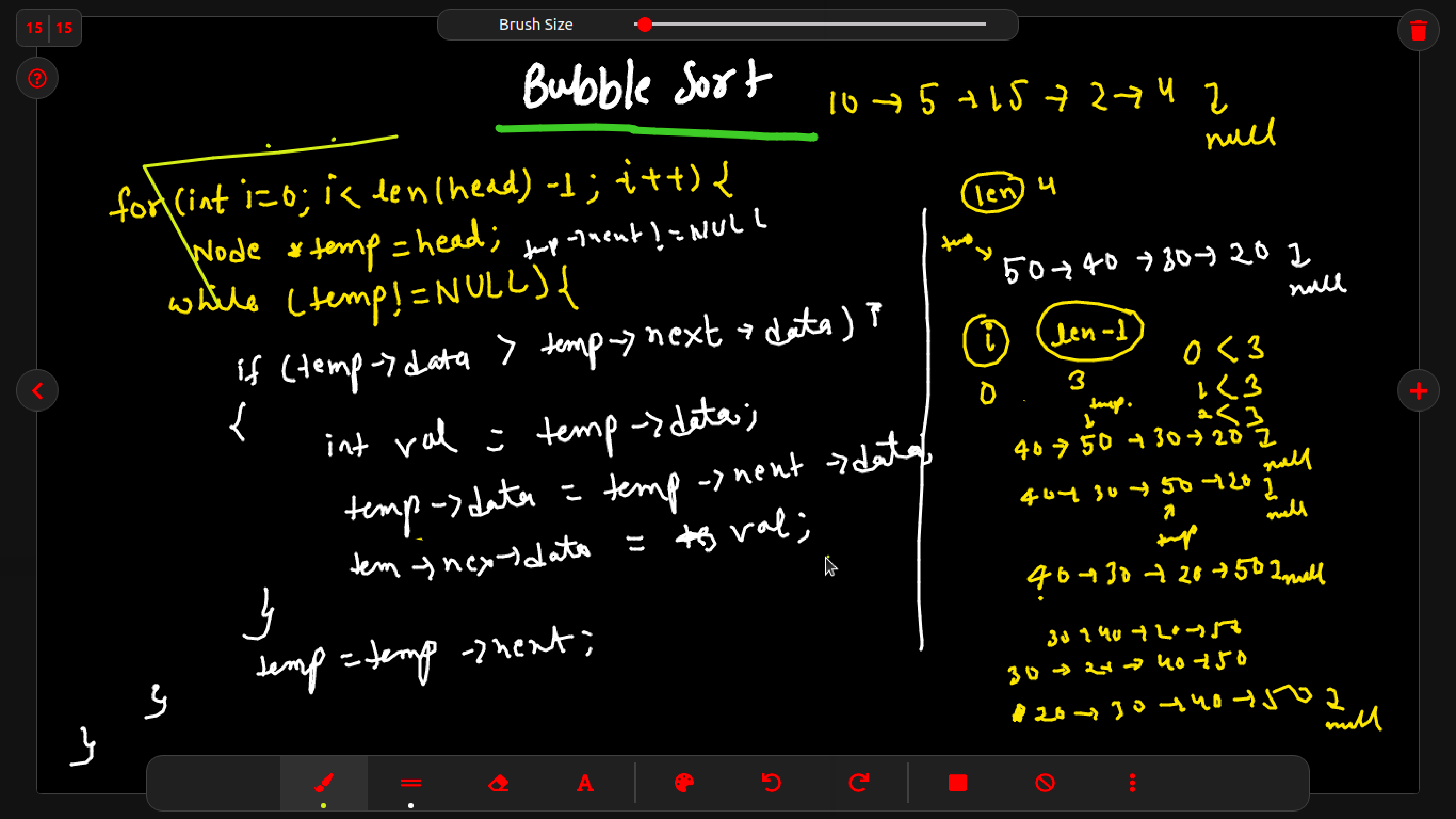
2

3 prev->next = temp

```
while (temp != NULL) {
    int count = 0;
    while (count < M) {
        count++;
        if (temp == NULL) return head
        prev = temp;
        temp = temp->next;
    }
    while (count < (m+n)) {        prev=NULL
        count++;
        if (temp == NULL) return head;
        temp = temp->next;
    }
```

Delete every N Nodes

```
Node* skipMdeleteN (Node *head, int M, int N){
    if(M==0)
        return NULL;

    Node *temp = head;
    Node *prev = NULL;
    while ( temp == NULL){
        int count =0;
        while (count <M) {
            if ( temp ==NULL)
                return head;

            count++;
            prev = temp;
            temp = temp->next;
        }
        while (count <M+N){
            if (temp==NULL) {
                prev=NULL;
                return head;
            }
            count++;
            temp =temp->next
        }
        prev -> next =temp
    }
    return head;
}
```



Bubble Sort

$10 \rightarrow 5 \rightarrow 15 \rightarrow 2 \rightarrow 4 \rightarrow$ null

```
for (int i=0; i< len(head) -1 ; i++) {
    Node *temp = head;        temp->next != NULL
  while (temp != NULL) {
      if (temp->data > temp->next->data) ?
      {
          int val = temp->data;
           temp->data = temp->next->data;
           tem->nex->data = ~~t~~ val;
      }
      temp = temp->next;
  }
}
```

$\boxed{\text{len}}$ 4

temp → $50 \rightarrow 40 \rightarrow 30 \rightarrow 20 \rightarrow$ null

$\boxed{i}$ $\boxed{\text{len}-1}$

0 3

$0 < 3$

$1 < 3$

$2 < 3$

temp

$40 \rightarrow 50 \rightarrow 30 \rightarrow 20 \rightarrow$ null

$40 \rightarrow 30 \rightarrow 50 \rightarrow 20 \rightarrow$ null

temp

$40 \rightarrow 30 \rightarrow 20 \rightarrow 50 \rightarrow$ null

$30 \rightarrow 40 \rightarrow 20 \rightarrow 50$

$30 \rightarrow 20 \rightarrow 40 \rightarrow 50$

$20 \rightarrow 30 \rightarrow 40 \rightarrow 50 \rightarrow$ null