Linked list - 1

Important question

(I) Delete a Node (recursive)

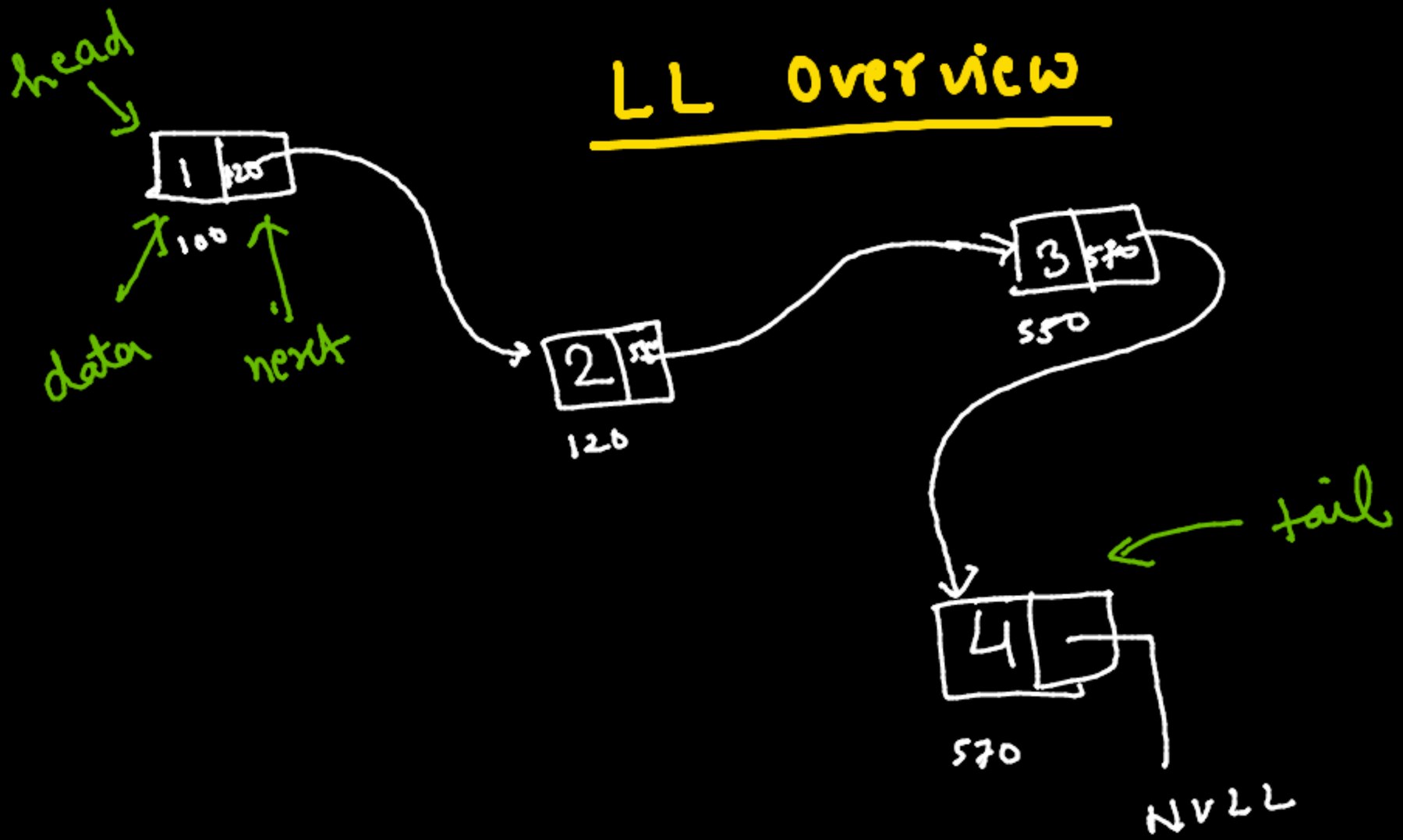
(II) Insert a Node (recursive)

(III) Append Last N To First

(IV) Eliminate duplicate

(V) Reverse a LL (iterative).

LL Overview



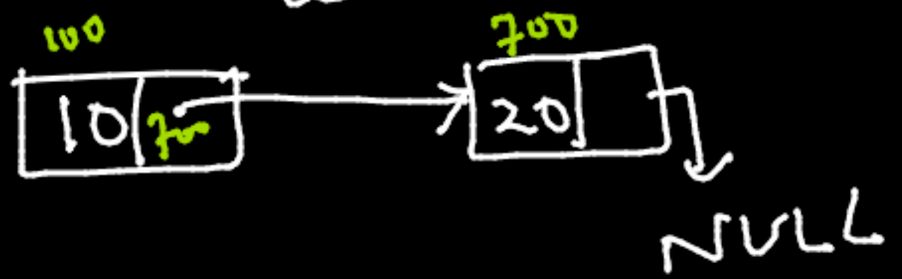
LinkedList

class Node {

```
class Node {

    int data;

    Node *next;

    Node (int data) {
    this -> data = data;
     next = NULL
    }

};
```

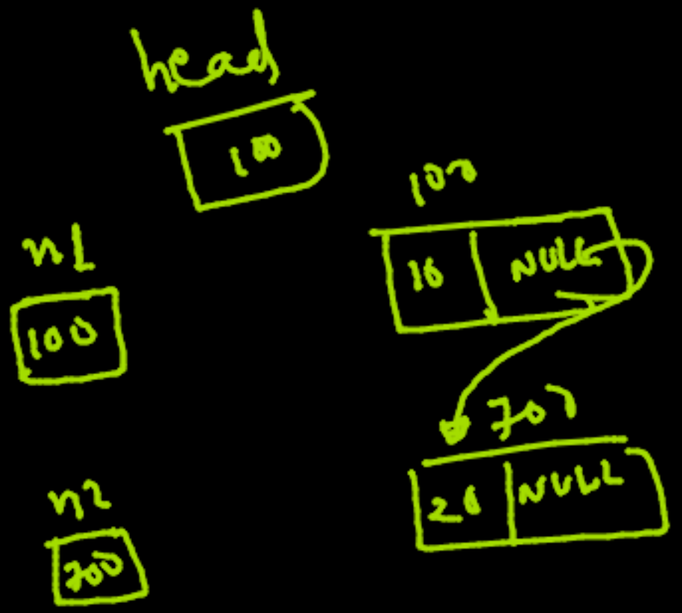
// Creating two node and connect them. (statically)

```
Node n1(10);

Node n2(20);

n1.next = n2;
Node *head = &n1;
```



Dynamic Allocation

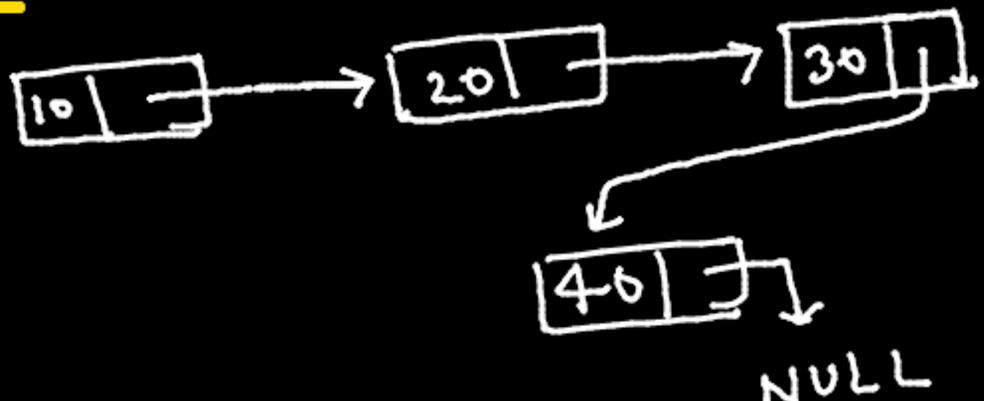
```
Node *n1 = new Node(10)
Node *n2 = new Nod(20)

n1->next = n2;
Node *head = n1.

cout << n1->data << "\n";
cout << n2->data << endl;
```



Print LL

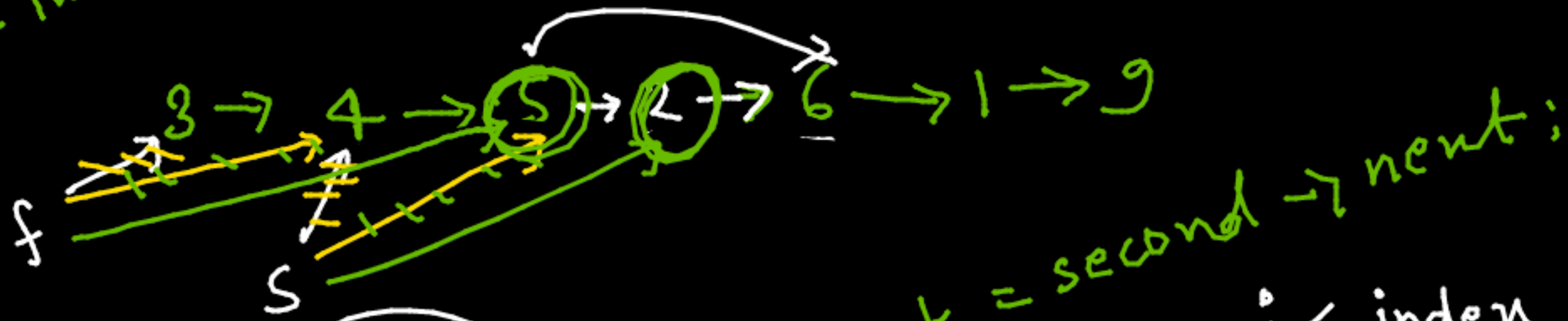
```
void printLL( Node *head) {
    Node *temp= head;
    while (temp != NULL){
        cout << temp->data <<" ";
        temp = temp ->next;
    }
}
```

10 20 30 40



Delete Node (given index)

eg- index = 3 i = 1



first->next = second->next;

return head;

i < index

1 < 3

2 < 3

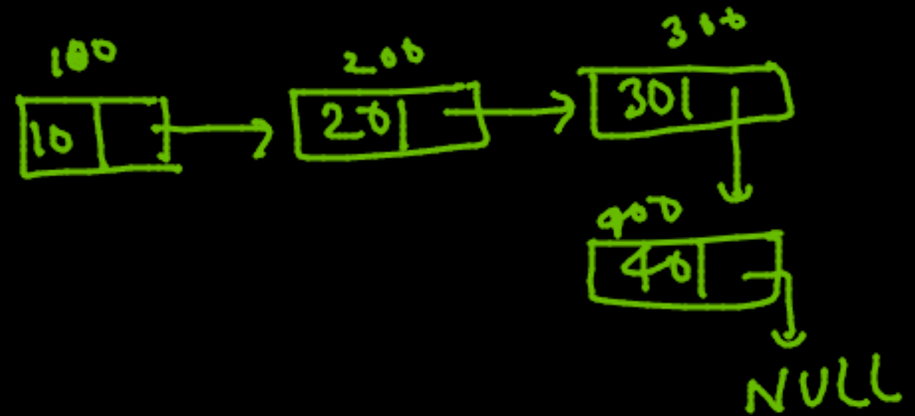
↓ Code

```
Node* delete (Node *head, int pos){
    if (pos >= len(head))
        return head;

    if (pos == 0)
        return head->next;

    int i=1;
    Node *first = head;
    Node *second = head->next;
    while (i< pos){
        first = first->next;
        second = second->next
    }

    first->next = second->next;
    return head;
}
```

suppose it's gives length of LL

Length of LL (Recursive)

```
int length (Node *head) {
    int count = 0;
    return helper (head, count);
}
int helper (Node *head, int count) {
    if ( head == NULL)
        return count;
    return helper (head->next, count+1);
}
```



head	count
~~100~~	~~0~~ 1
~~200~~	~~1~~ 2
~~300~~	~~2~~ 3
~~400~~	4
NULL	

Insert a Node (ith pos)

↳ iterative

data 100 pos 0

At start

10 → 20 → 30 → 40

head → [100] →

```
Node *n1 = new Node(data);

if (pos == 0) {
    n1->next = head
    head = n1;
    return head;
}
```

for any pos

At any pos. pos = 2
data = 100


```
Node *newNode = new Node(data);
int i=1; Node *temp=head
while (i<pos) {
    prev = temp; temp=temp->next; i++
}
newNode->next = prev->next;
prev->next = newNode
return head
}
```

1 < 2

Insert a Node (Recursive)

0 < i

eg. (i) (data)
2 100

head, i, data
given


```
Node* insert(Node *head, int i, int data){
    //base case
    if (i==0 || head == NULL)
        return head;

    Node *temp = head;
    Node *newNode = new Node(data);
    int count=0;
    helper(temp, i, newNode);

}
```

count

```
void helper(Node *temp, int i, Node *newNode, int count){
    if (count < i-1){
        helper(temp->next, i, newNode, count+1);
    }else{
        newNode->next = temp->next;
        temp->next = newNode;
        return;
    }
}
```

if (temp != NULL) return;

for return > null left it

Ind. Delete Recursive AMode

1 → 2 → 3 → 4 → 5

Testcase

1 → 2 → 3 → 4 → NULL

pos 5

0 to 5-1

index < pos-1

0 < 4

1 < 4

2 < 4

3 < 4

Append LastN To First

1 2 3 4 5 (len) 5

3

3 → 4 → 5 → 1 → 2 → NULL

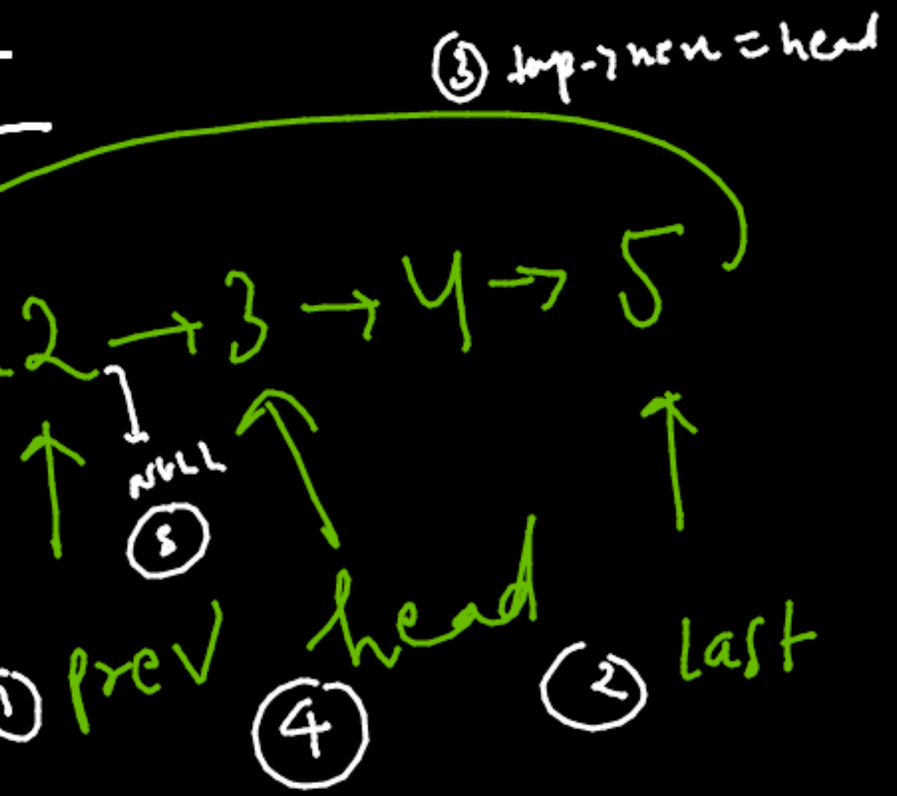
10 → 20 → 30 → 40 → 50 → 60 (len) 6

4

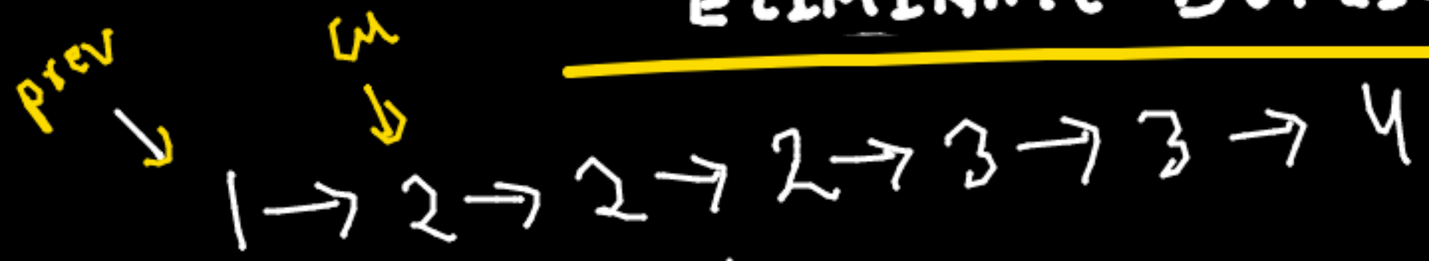
⇒ 30 → 40 → 50 → 60 → 10 → 20 → NULL

5 - 3 = 2

6 - 4 = 2



ELIMINATE DUPLICATE



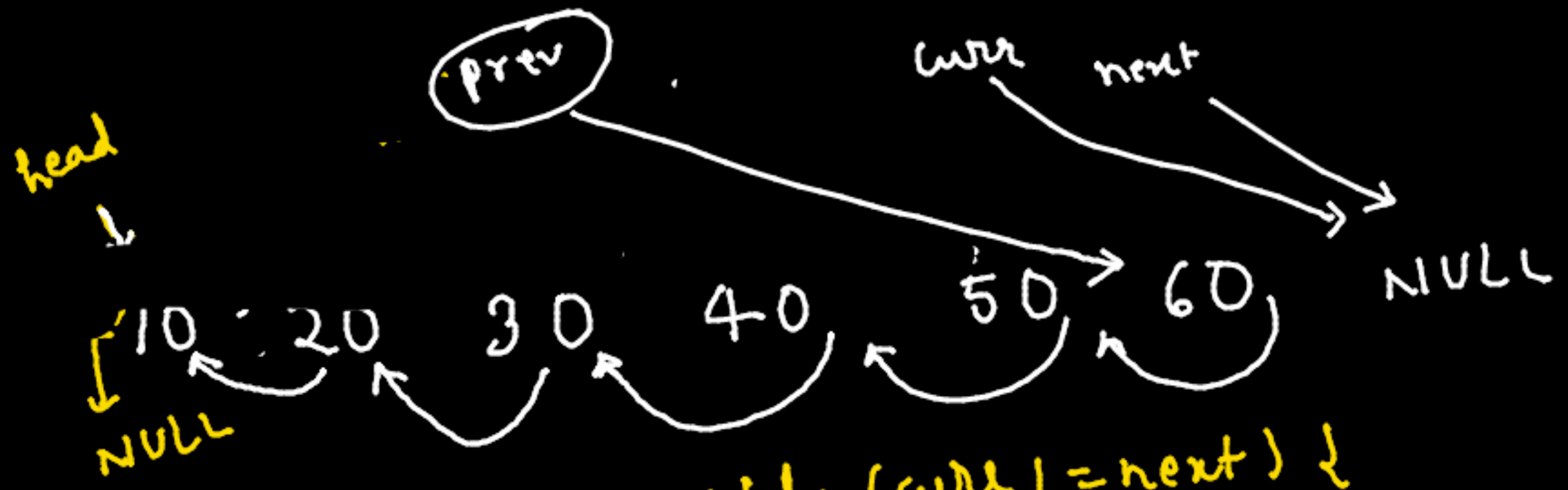

```
if (head==NULL) return;
Node *curr = head->next;
Node *prev = head;
while (curr!=NULL)
  if (curr->data== prev->data && (curr->next!=NULL)
      curr = curr->next;
else  if (curr->next==NULL && prev->data== curr->data)
          prev->next=NULL; return head;
      else   prev->next =curr, prev = curr, curr = prev->next;
```

Reverse a LL


```
Node *curr = head;
Node *prev = NULL;
Node *next = NULL;
```

```
while (curr != NULL) {
    next = curr -> next;
    curr -> next = prev;
    prev = curr;
    curr = next;
}
head = prev;
return head;
```

More Clear




```
Node *curr = head;
Node *prev = NULL;
Node *next = NULL
```

```
while (curr != next) {
  next = curr->next
  curr->next = prev;
  prev = curr;
  curr = next;
}
head = prev;
retur head;
```

palindrome bhi same hi

Palindrome

$3 \to 9 \to 1 \to 2 \to 9 \to 3$

reverse and check

$3 \to 9 \to 2 \to 1 \to 9 \to 3$