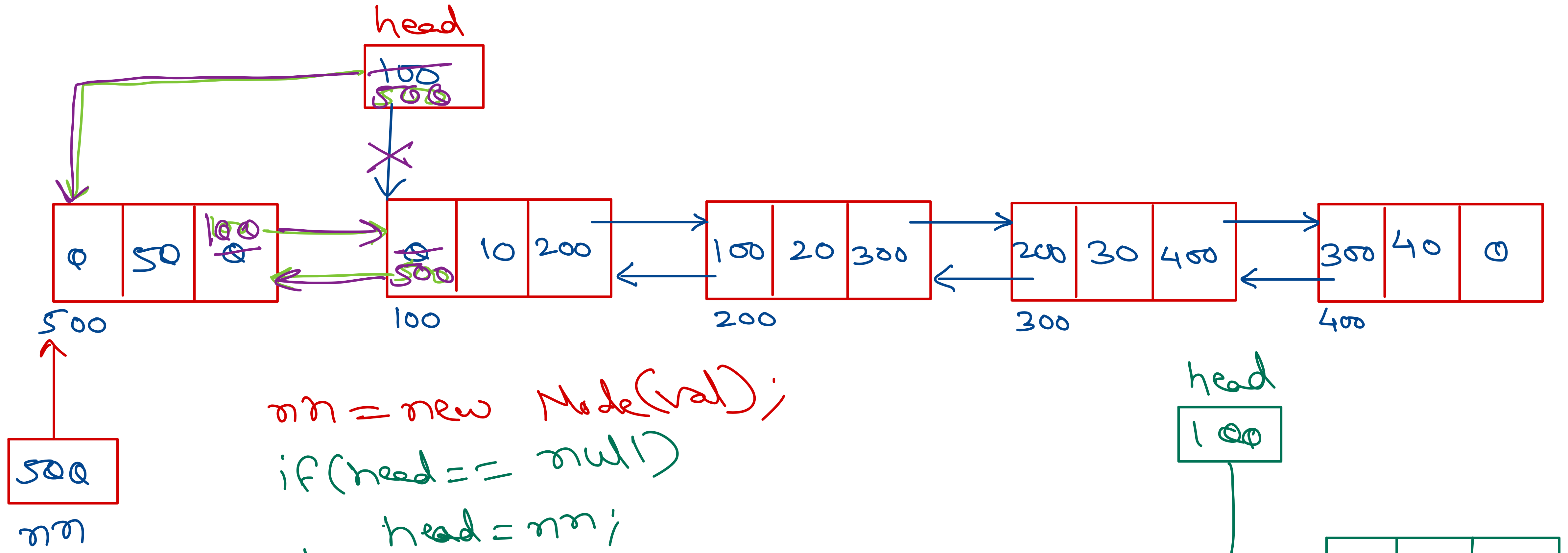
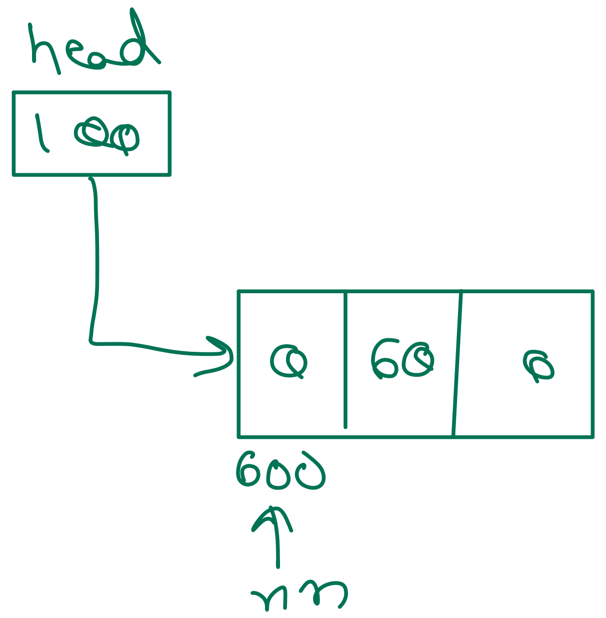
Data Structure & Algorithms

Nilesh Ghule

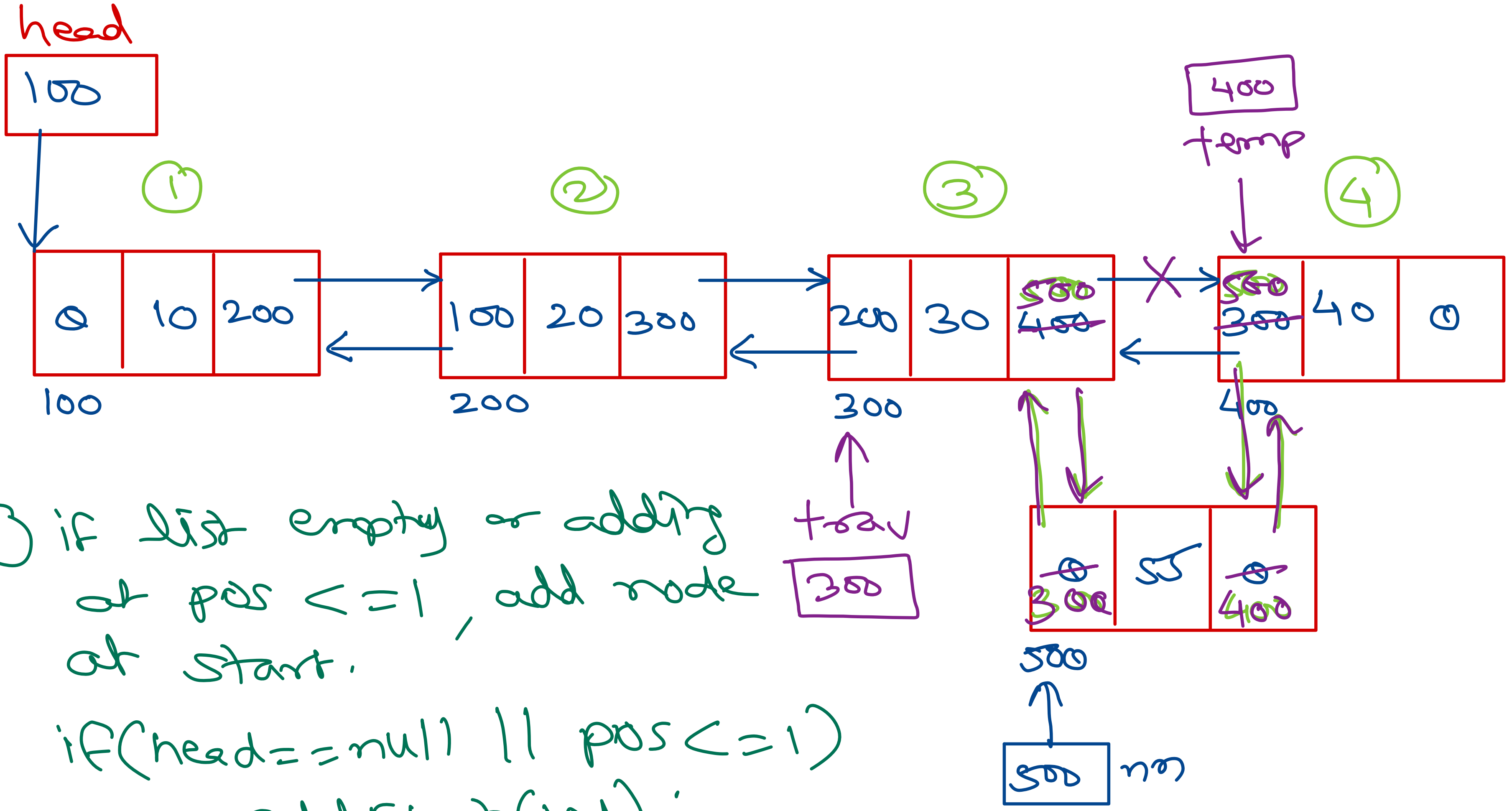
Doubly Linear Linked List - addFirst()



```
nn = new Node(val);
if(head == null)
        head = nn;
else {
    ✓ nn.next = head;
    ✓ head.prev = nn;
    ✓ head = nn;
}
```



Doubly Linear Linked List - addAtPos()

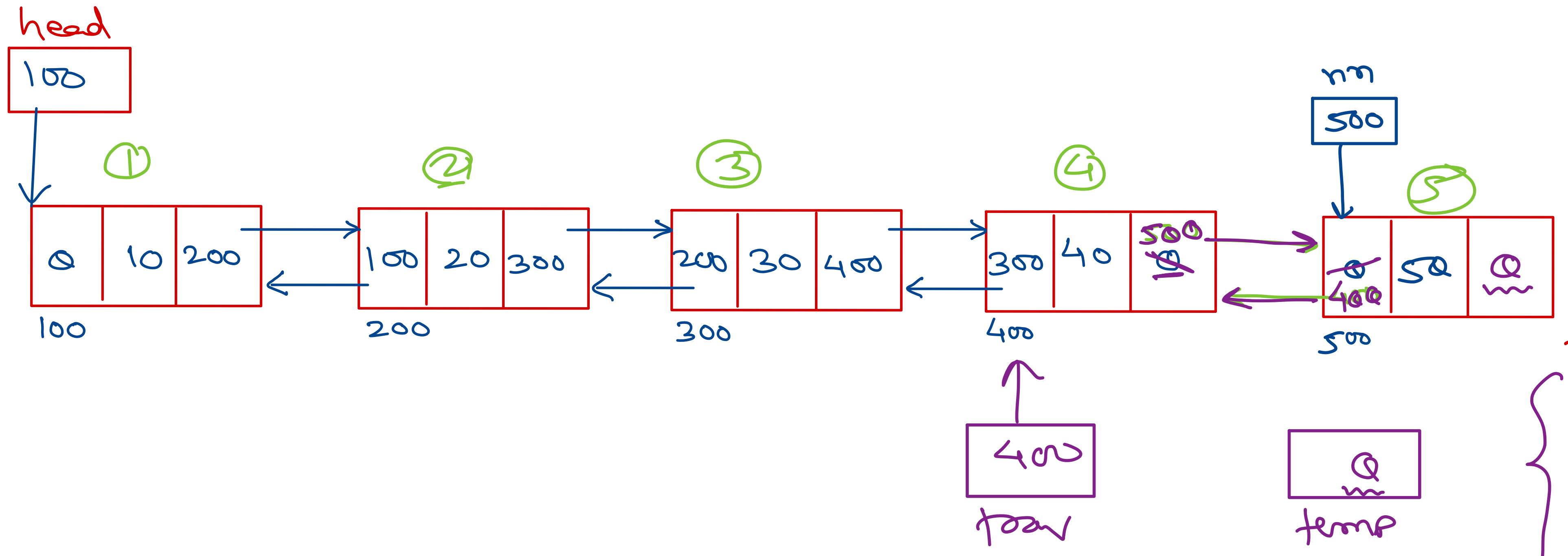


① if list empty or adding at pos <=1, add node at start.

```
if(head==null || pos<=1)
        addFirst(val);
```

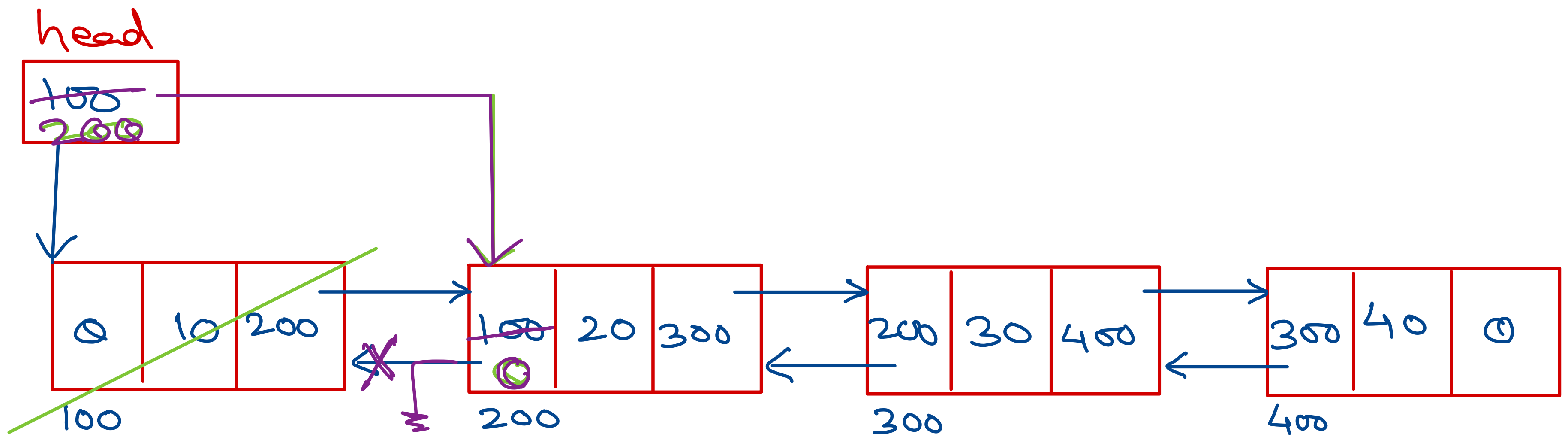
```
nn=new Node(val);
trav=head;
for( i=1; i<pos-1;i++)
    trav=trav.next;
temp=trav.next;

nn.next = temp;
nn.prev= trav;
trav.next =nn;

temp.prev= nn;
```

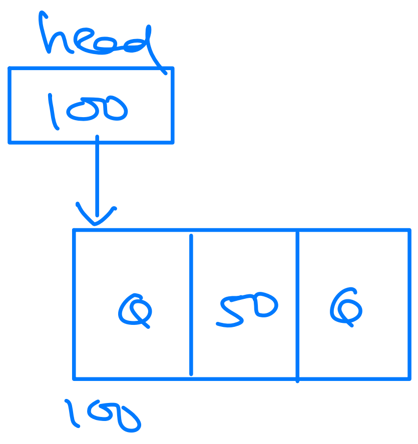
Doubly Linear Linked List - addAtpos() - special case = last pos. ②



```
nn = new Node(val);
trav = head;
for( i=1; i < pos-1; i++){
    if(trav.next == null)
        break;
    trav = trav.next;
}
✓ temp = trav.next;
✓ nn.next = temp;
✓ nn.prev = trav;
✓ trav.next = nn;
  if(temp != null)
NPE ✗ temp.prev = nn;
```

Doubly Linear Linked List – del First()





✓

head = head.next;
head.prev = null;

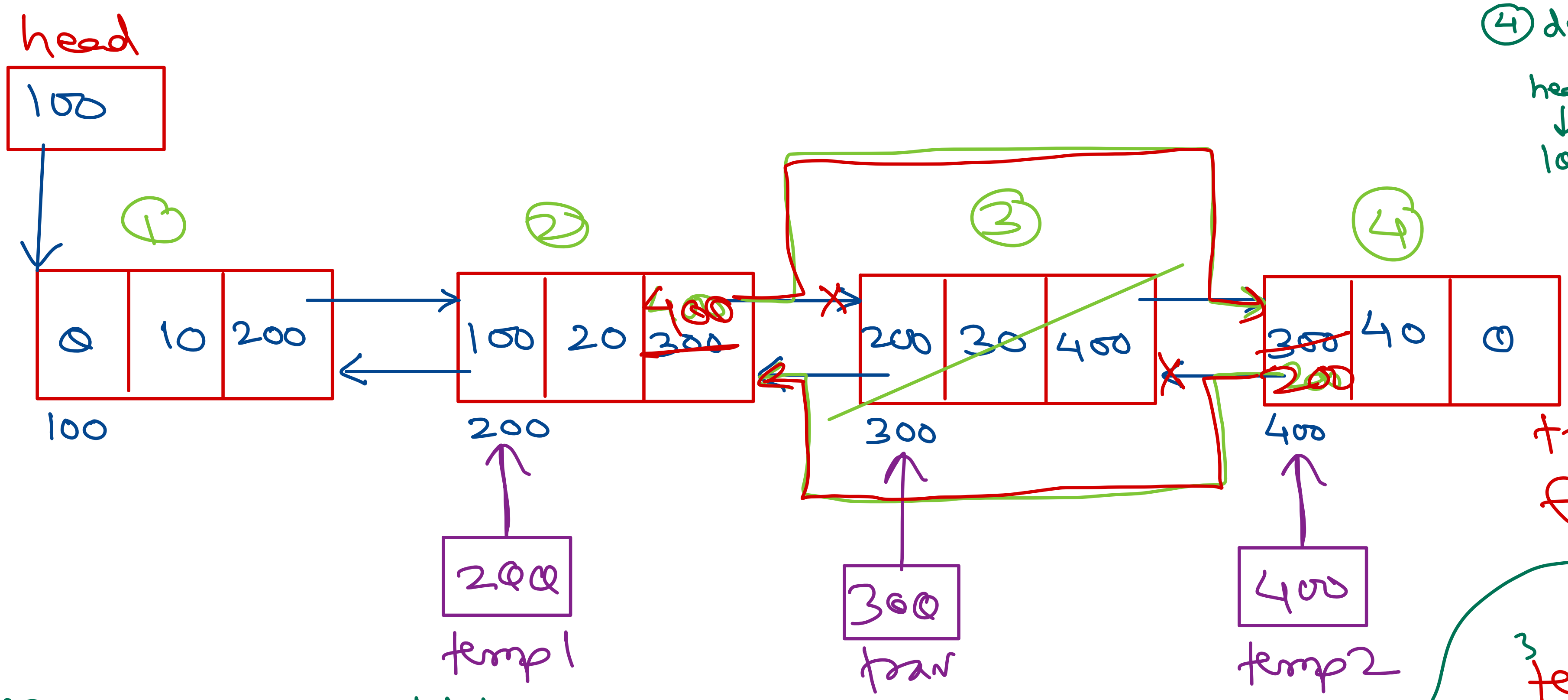
if list is empty, do nothing.

if (head == null)
    return;

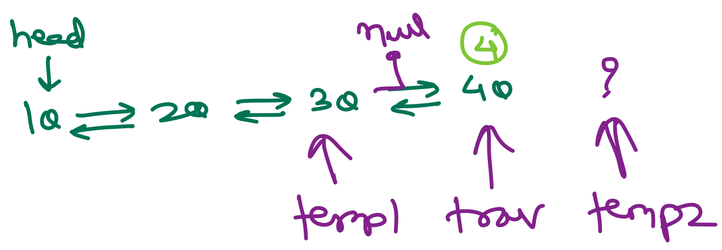
if list has single node, then delete it.

if(head.next == null)
    head = null;

Doubly Linear Linked List – delAtPos()



④ deleting the last node.



① if list empty or delete at first pos, call delFirst();

```
if(head==null || pos==1)
    delFirst();
```

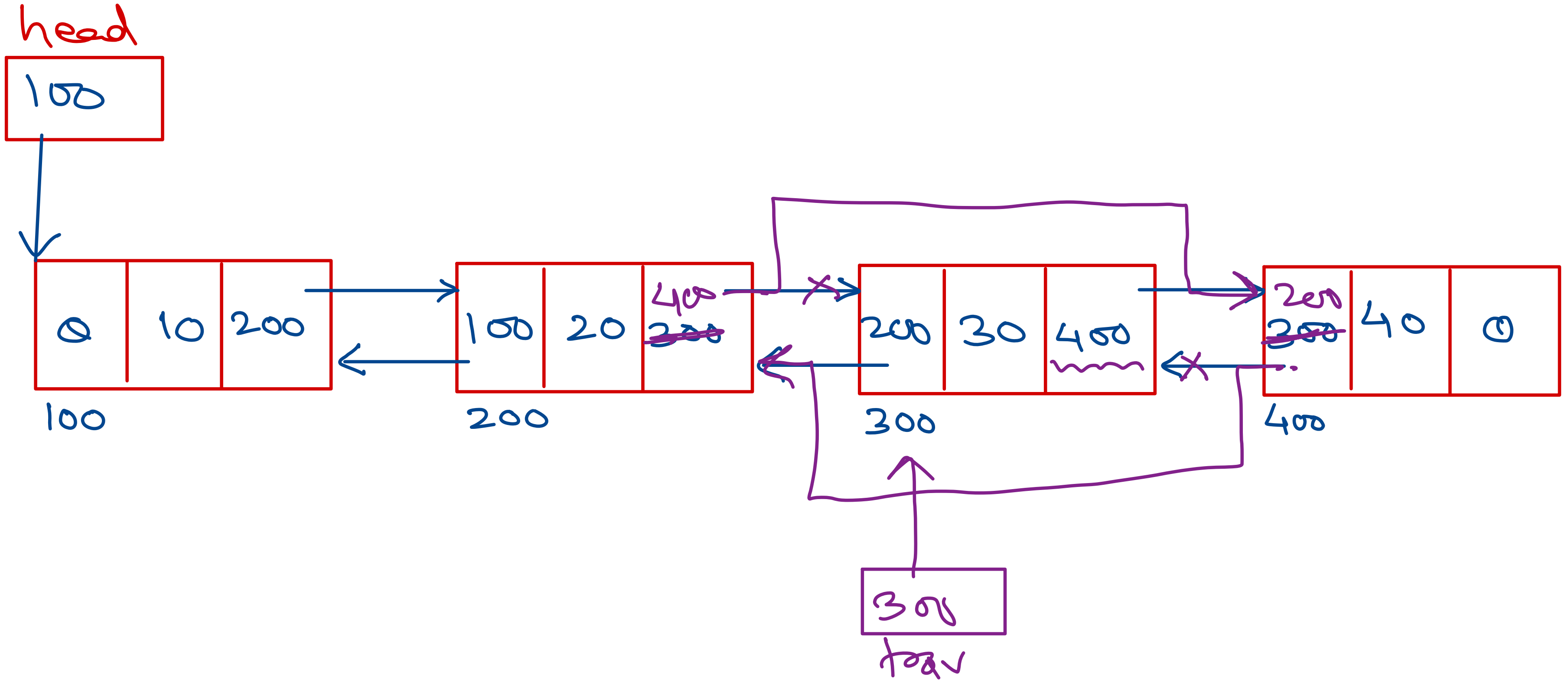
② if deleting at pos less than 1, do nothing.

```
if(pos < 1)
    return;
```

③ if deleting beyond last pos, then do nothing.

```
trav = head;
for(i=1; i<pos; i++){
    if(trav.next==null)
        return;
    trav = trav.next;
}
temp1 = trav.prev;
temp2 = trav.next;

✓ temp1.next = temp2;
if(temp2 != null)
    ✓ temp2.prev = temp1;
```

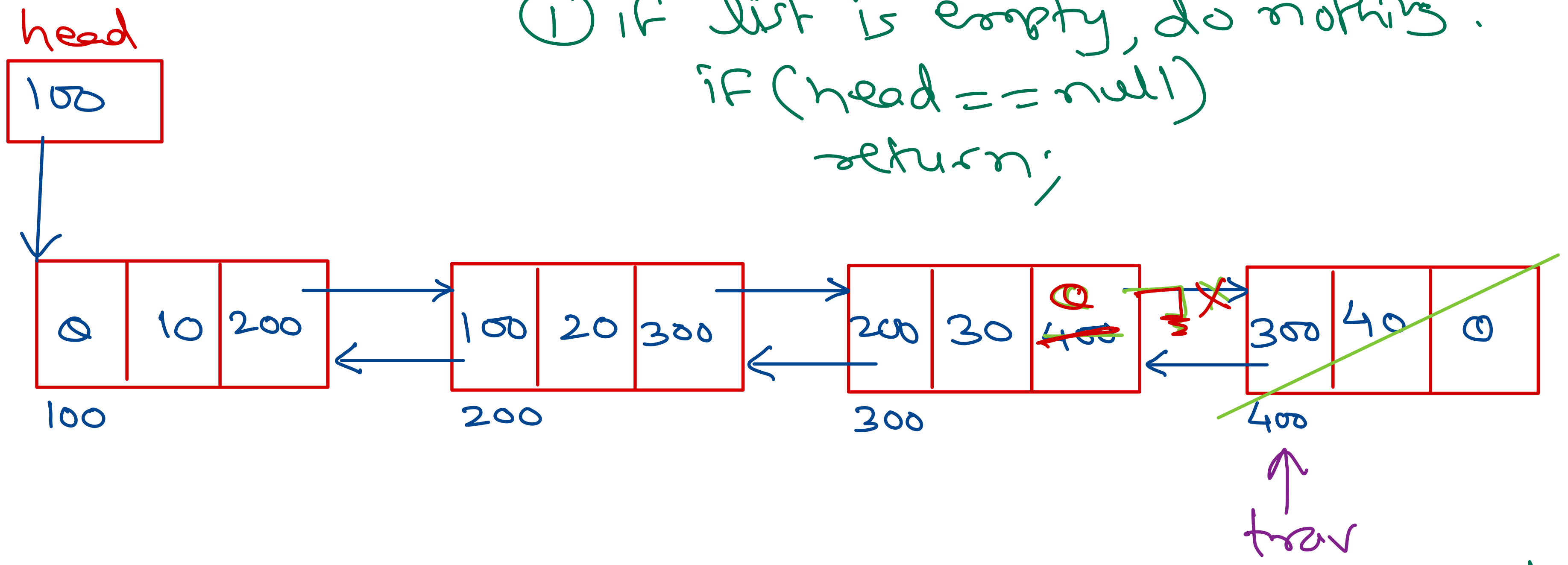
Doubly Linear Linked List - delAtPos()



trav.prev.next = trav.next;

trav.next.prev = trav.prev;

Doubly Linear Linked List - del Last()



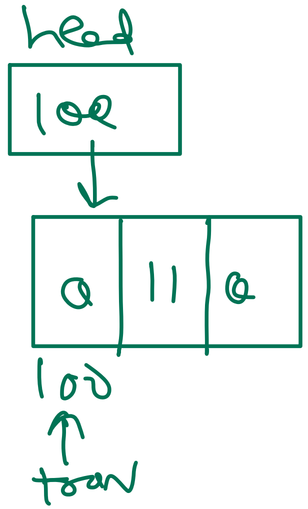
① if list is empty, do nothing.

```
if(head == null)
    return;
```

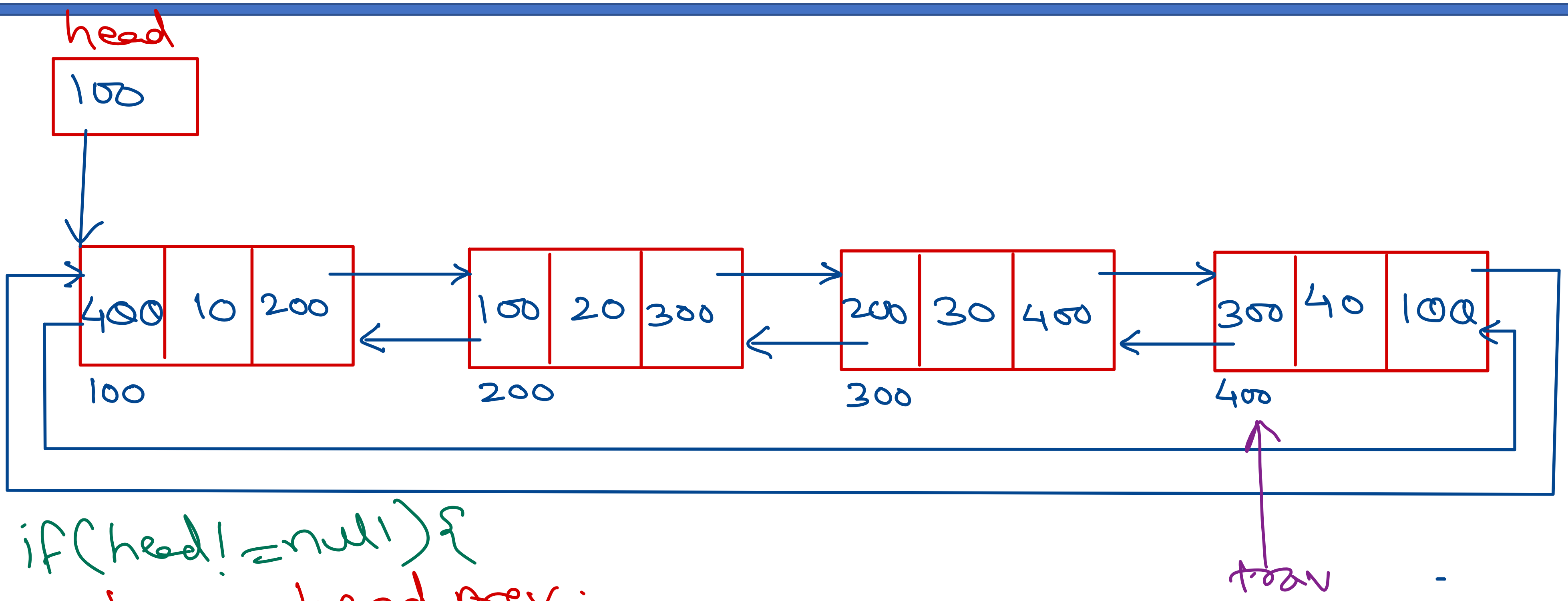
```
trav = head;
while(trav.next != null)
    trav = trav.next;

trav.prev.next = null;
```

② if list has single node, then delete it.

```
if(head.next == null)
    head = null;
```



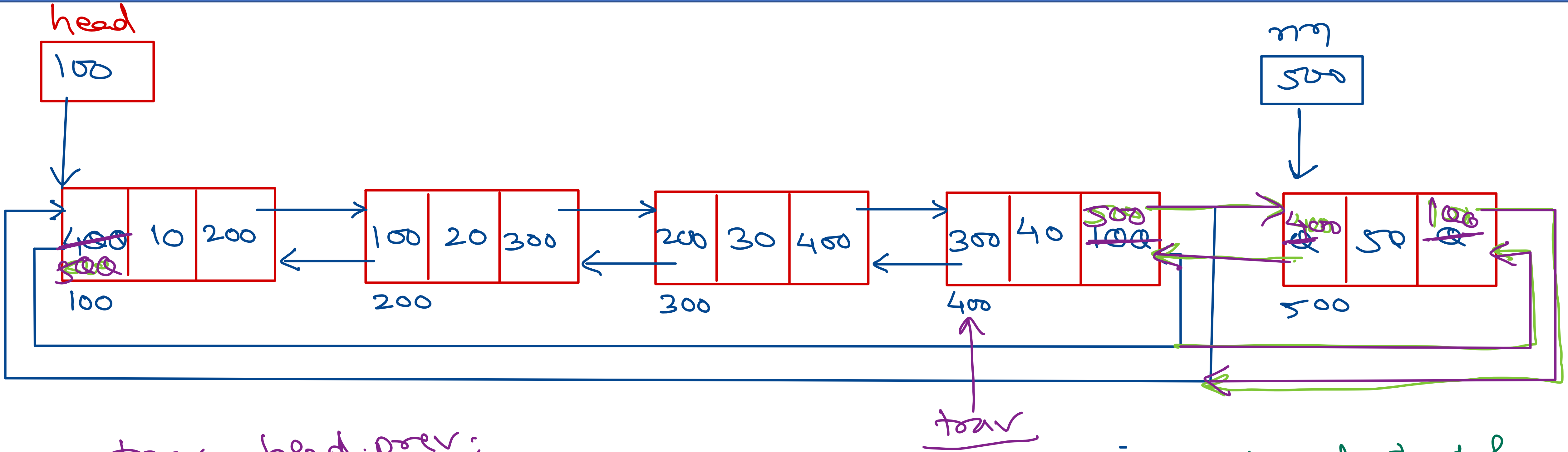
Doubly Circular Linked List – displayRev()



```
if(head!=null){
    trav= head.prev;
    do{
        print(trav.data);
        trav= trav. prev;
    }while(trav != head.prev);
}
```

40, 30, 20, 10

Doubly Circular Linked List – addLast()

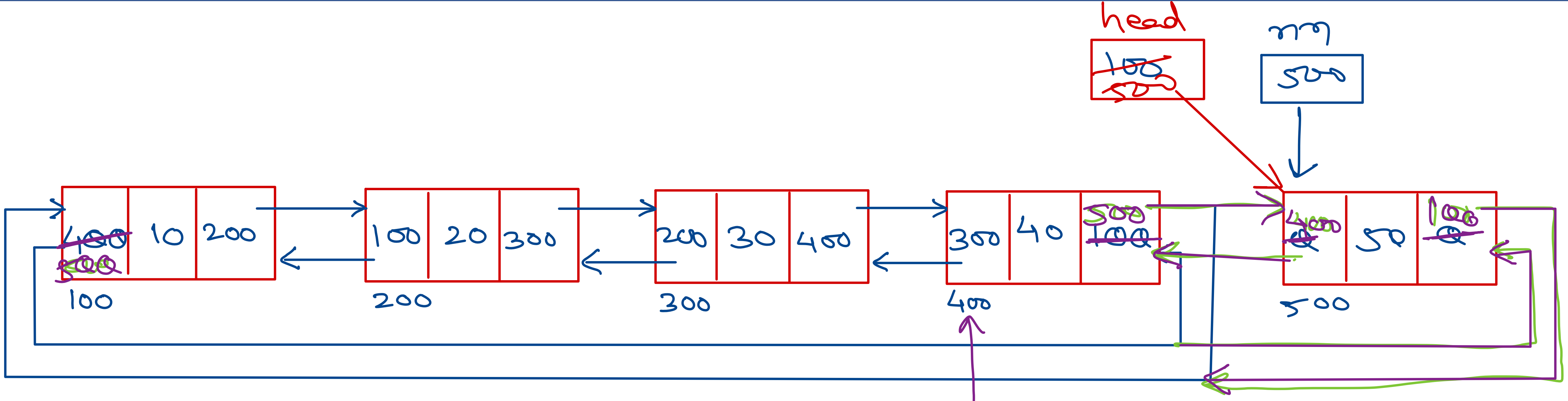


```
trav = head.prev;
nn.next = head;
nn.prev = trav;
trav.next = nn;
head.prev = nn;
```

① if list empty, add node at start & make it circular.

```
if(head==null) {
    head = nn;
    nn.next = head;
    nn.prev = head;
}
```

Doubly Circular Linked List – addFirst ( )



```
trav = head.prev;
nn.next = head;
nn.prev = trav;
trav.next = nn;
head.prev = nn;
head = nn;
```

① if list empty, add node at start & make it circular.

```
if(head==null) {
    head = nn;
    nn.next = head;
    nn.prev = head;
}
```

Linked List – Competitive programming -selection sort

- Sort the singly linked list.

$$\begin{tikzcd}
\text{head} \arrow[d] & & & & & \\
44 \arrow[r] & 22 \arrow[r] & 55 \arrow[r] & 11 \arrow[r] & 33 \arrow[r] & \perp
\end{tikzcd}$$

```
Node i, j;
for (i = head ; i != null ; i = i.next) {
    for (j = i.next ; j != null ; j = j.next) {
        if ( i.data > j.data ) {
            int t = i.data ;
            i.data = j.data ;
            j.data = t ;
        }
    }
}
```

Linked List – Competitive programming

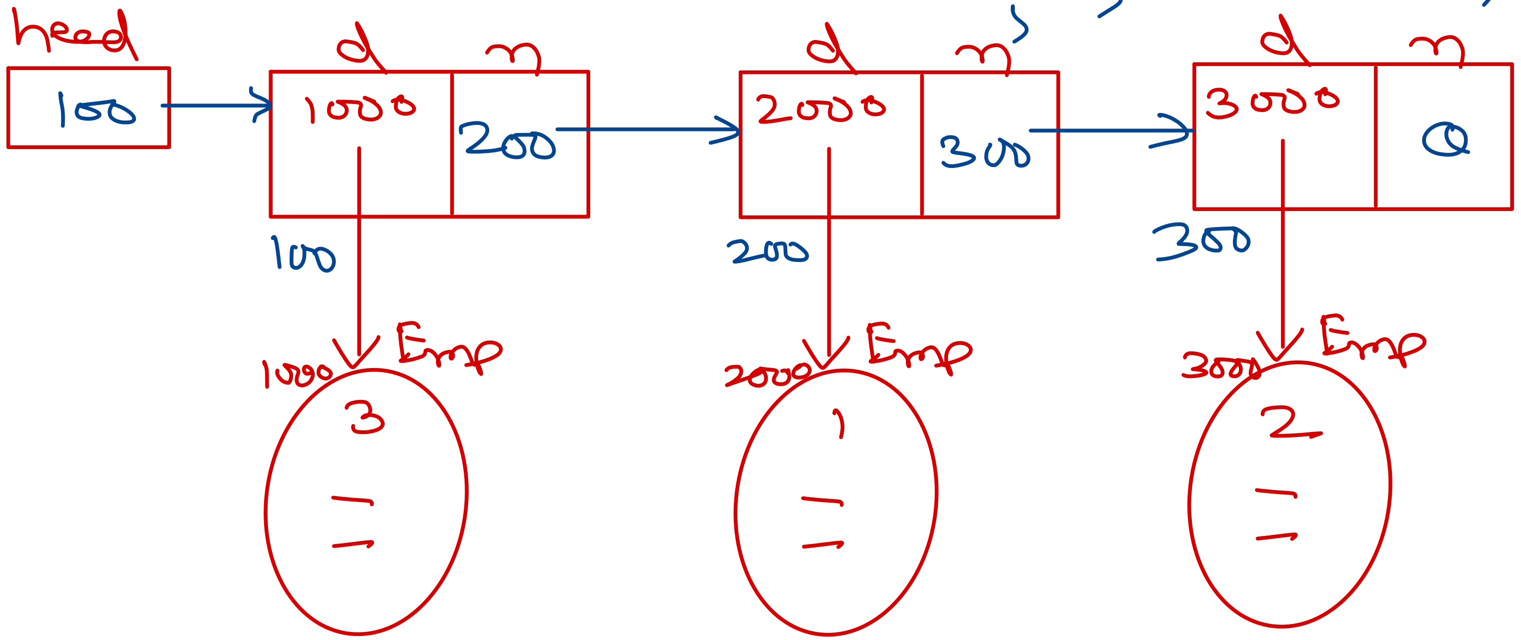
- Sort the singly linked list.

```
class Emp {
    int id;
    String name;
    double sal;
    =
}

class List {
    static class Node {
        Emp data;
        Node next;
        =
    }
    private Node head;
    :
}
```

```
main() {
    List l=new List();
    l.addLast(new Emp(3, ...));
    l.addLast(new Emp(1, ...));
    l.addLast(new Emp(2, ...));
}
```

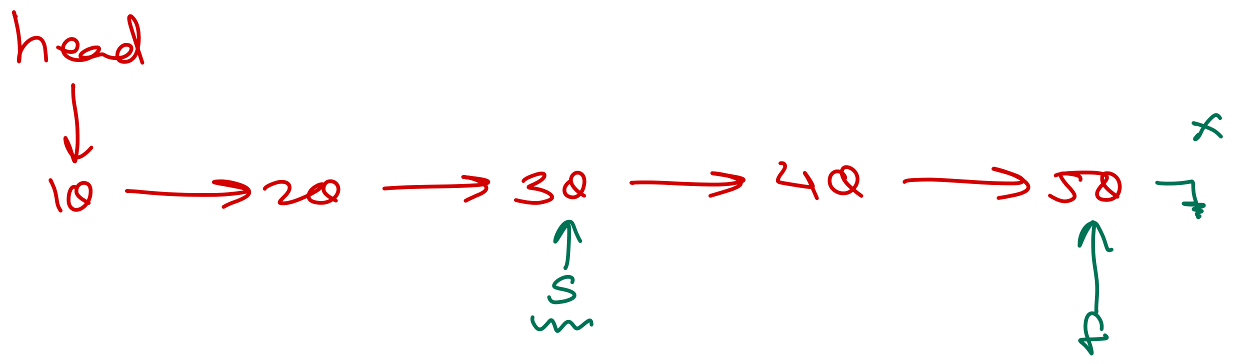
In List class:

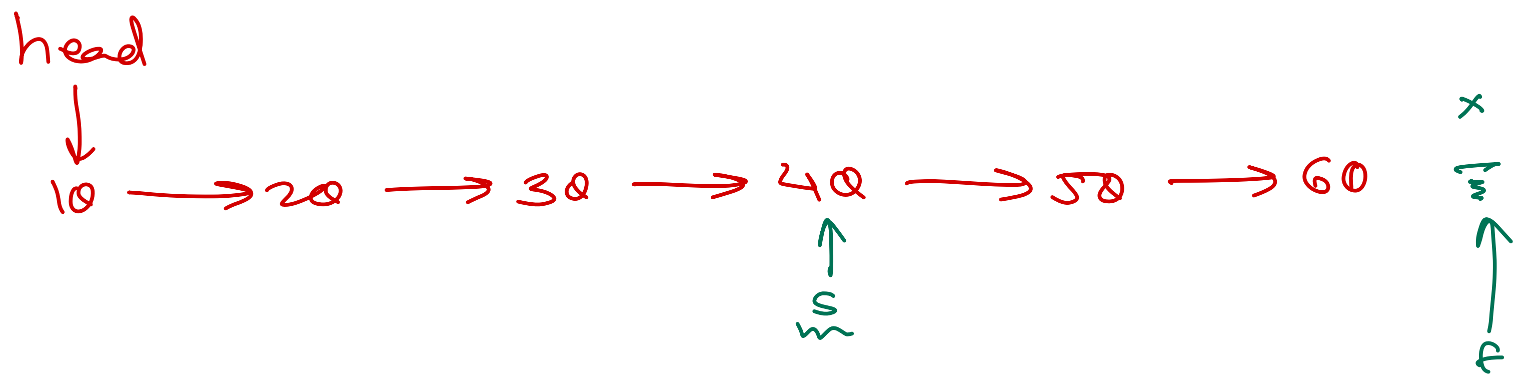
```
void addLast(Emp val) {
    Node nn=new Node(val);
    if(head==null)
        head=nn;
    else {
        Node trav=head;
        while(trav.next!=null)
            trav=trav.next;
        trav.next=nn;
    }
}
```



Linked List – Competitive programming

- Find middle of singly linear linked list.





SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>