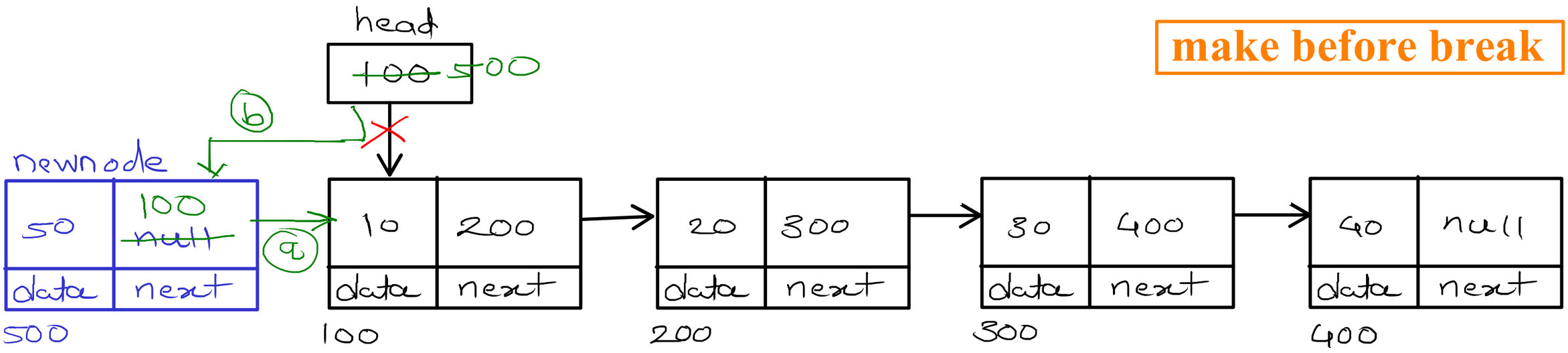
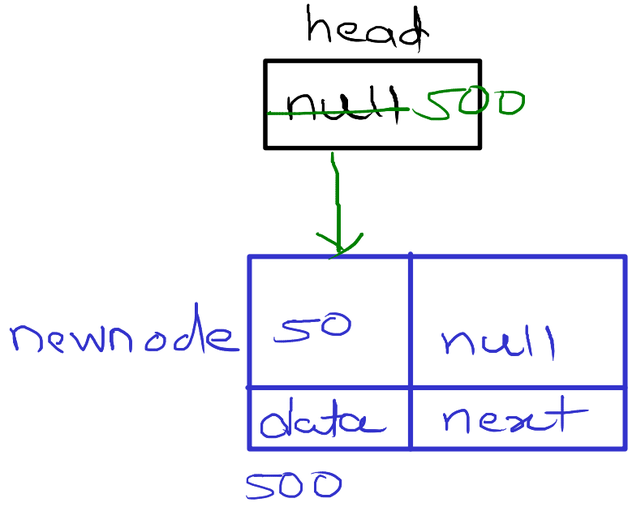
Singly Linear Linked List - Add First



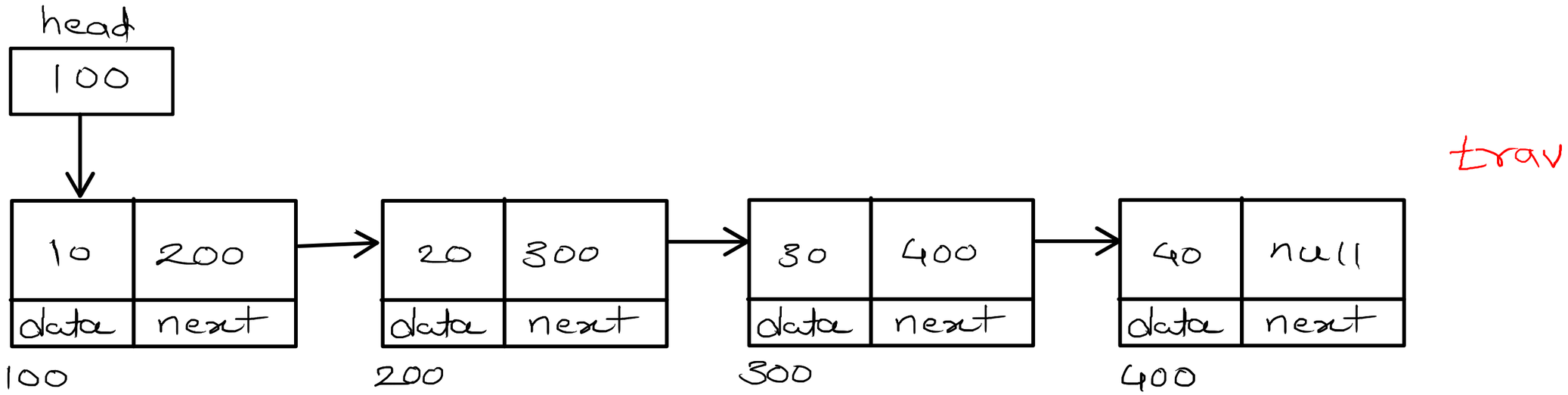
make before break




```
//1. create node with given value
//2. if list is empty
        //a. add newnode into head itself
//3. if list is not empty
        //a. add first node into next of newnode
        //b. add newnode into head
```

Time Complexity : O(1)

Singly Linear Linked List - Display



trav trav.data trav.next

100 10 200

200 20 300

300 30 400

400 40 null

null

//1. create trav and start at head

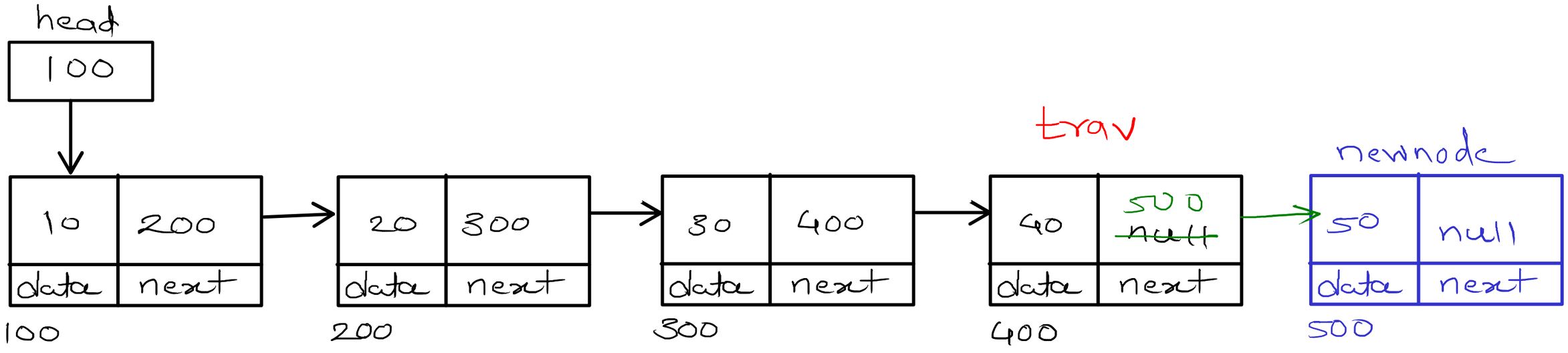
//2. print(visit) data of current node

//3. go on nexxt node

//4. repeat step 2 and 3 till last node

Time Complexity : O(n)

Singly Linear Linked List - Add Last

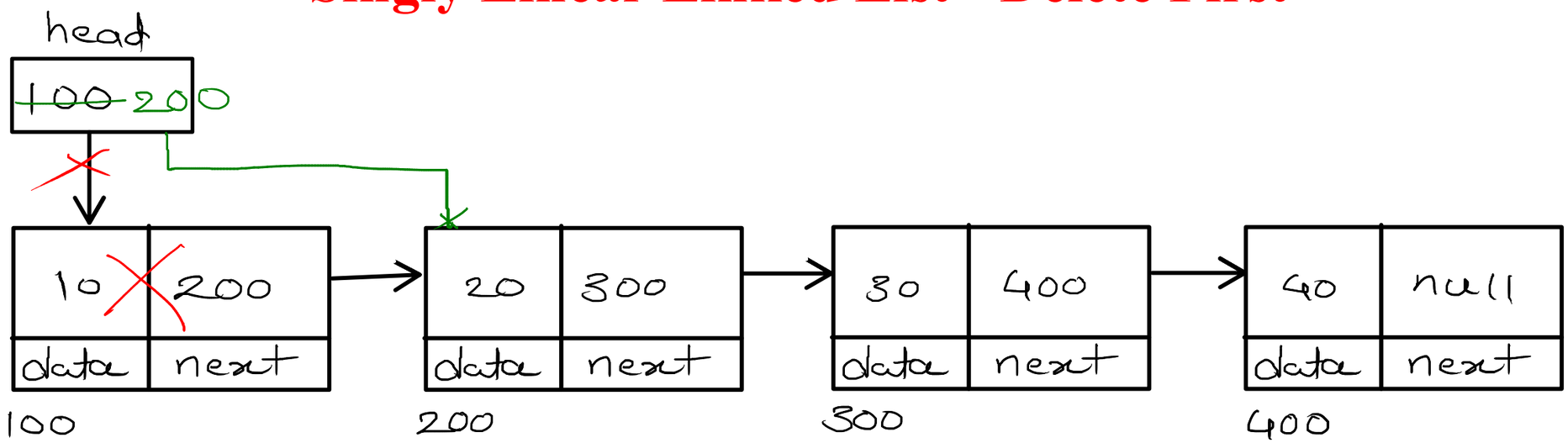


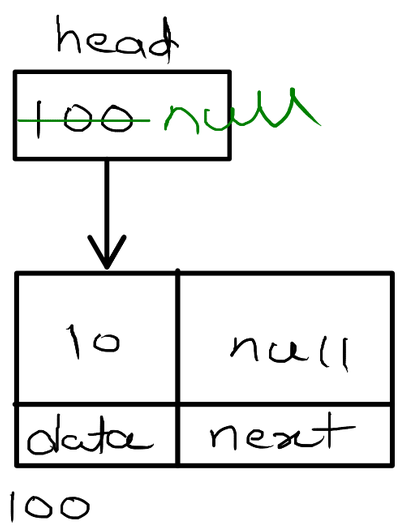

```
trav = head;
while(trav.next != null)
    trav = trav.next;
```

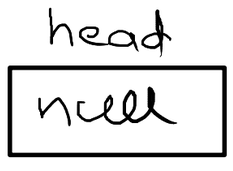
```
//1. create node with given value
//2. if list is empty
      //a. add newnode into head
//3. if list is not empty
      //a. traverse till last node
      //b. add newnode into next of trav
```

Time Complexity : O(n)

Singly Linear Linked List - Delete First



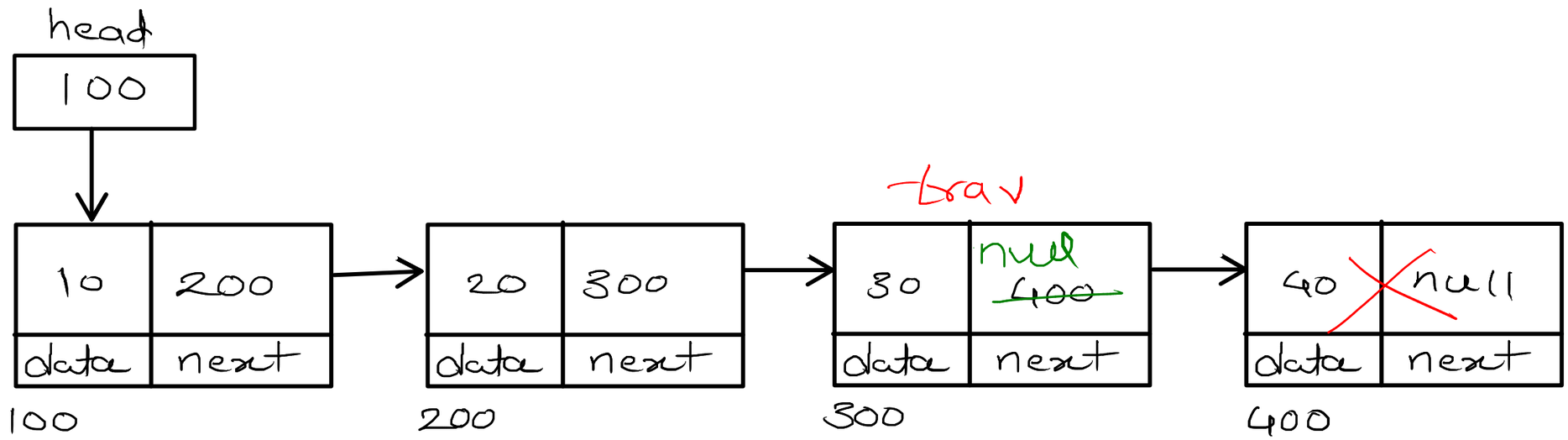


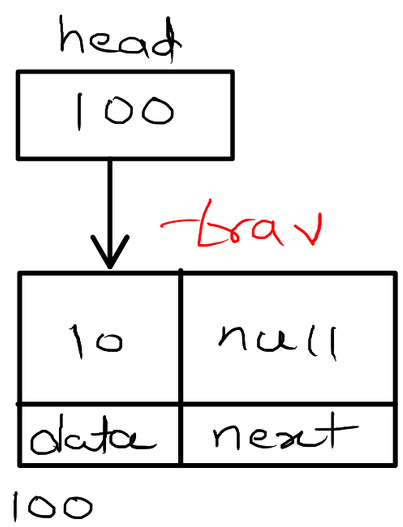



```
//1. if list is empty
      // return;
//2. if list is not empty
      //a. move head on second node
```

Time Complexity : O(1)

Singly Linear Linked List - Delete Last



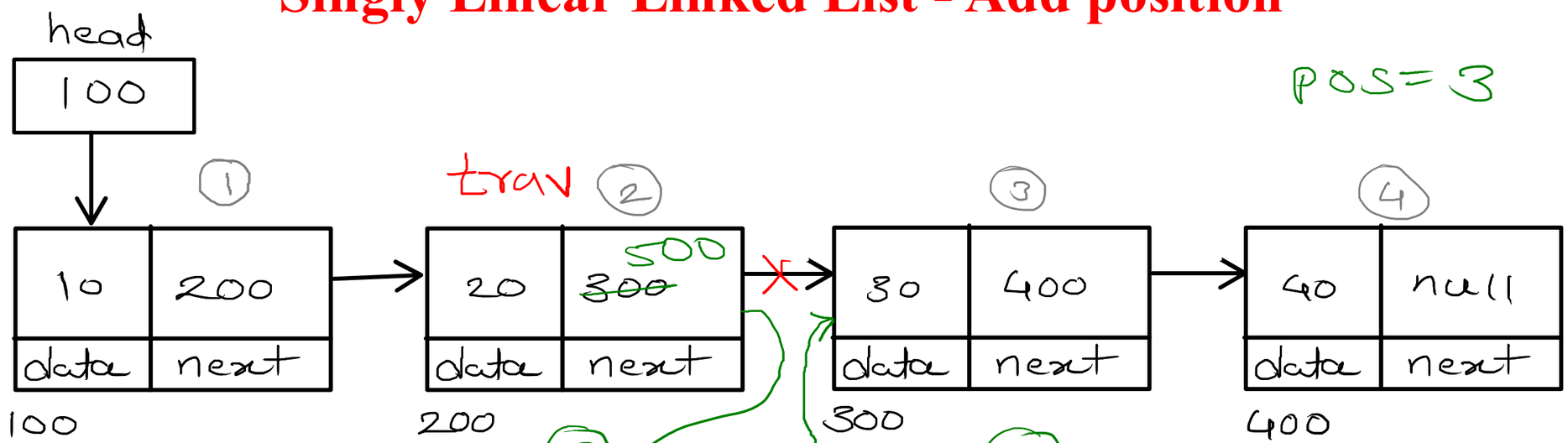



```
//1. if list is empty
        return;
//2. if list has single node
        //a. make head equal to null
//3. if list has multiple nodes
        //a. traverse till second last node
        //b. add null into next of second last node
```

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

Time Complexity : O(n)

Singly Linear Linked List - Add position




```
trav = head;
for(i=1; i<pos-1; i++)
    trav = trav.next;
```

pos = 3

trav	i	i<2
100	1	T
200	2	F

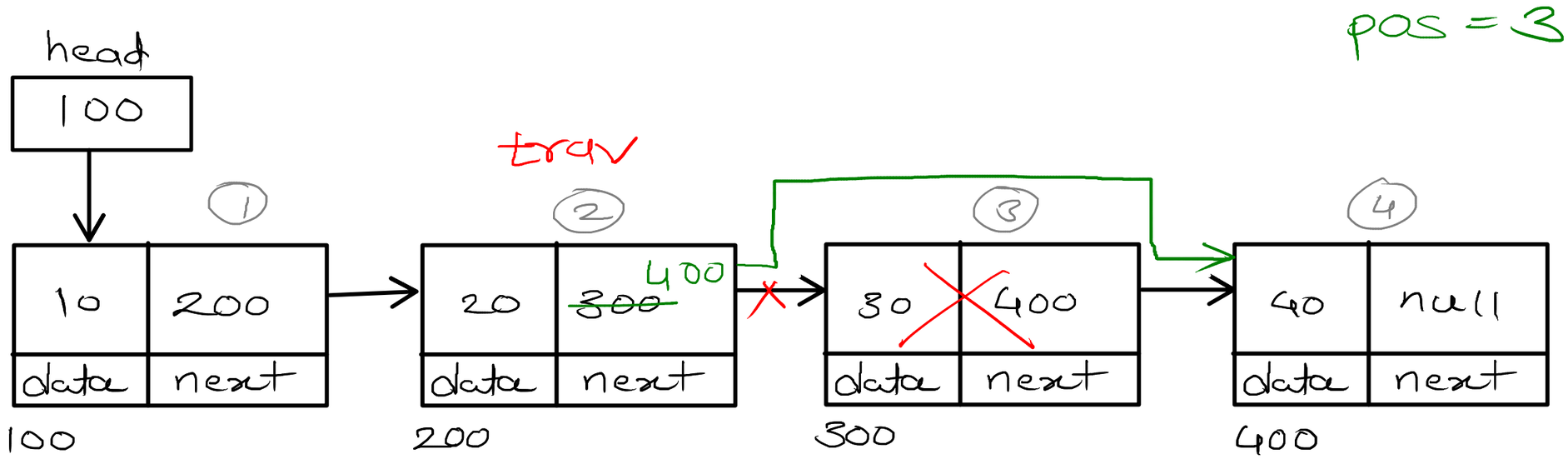
pos = 4

trav	i	i<3
100	1	T
200	2	T
300	3	F

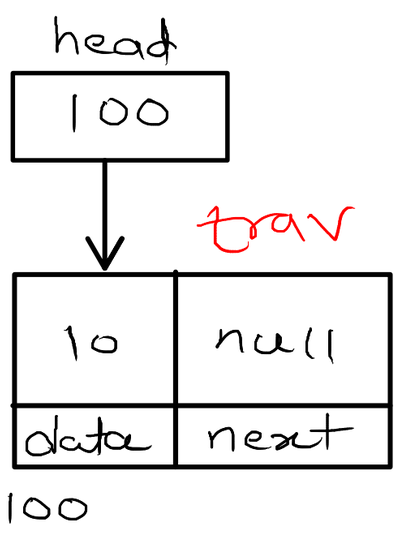
//1. create node with given data
//2. if list is empty
//a. add newnode into head
//3. if list is not empty
//a. traverse till pos-1 node
//b. add pos node into next of newnode
//c. add newnode into next of pos-1 node

Time Complexity : O(n)

Singly Linear Linked List - Delete position



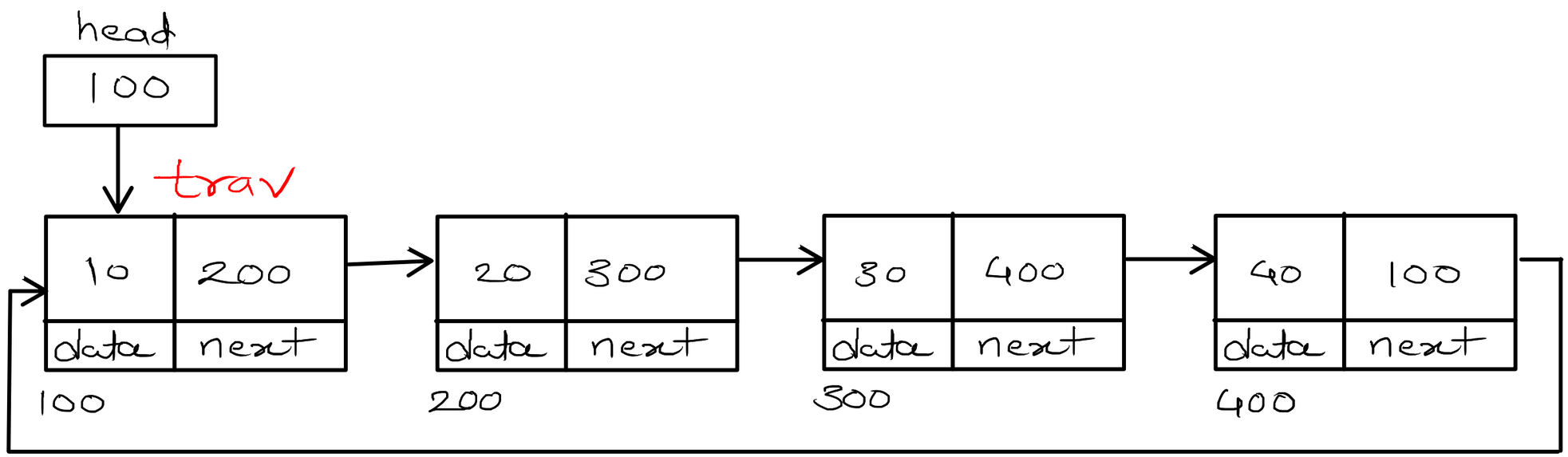

```
//1. if list is empty
        return;
//2. if list is not empty
        //a. traverse till pos-1 node
        //b. add pos+1 node into next of pos-1 node
```



trav.next = trav.next.next

Time Complexity : O(n)

Singly Circular Linked List - Display

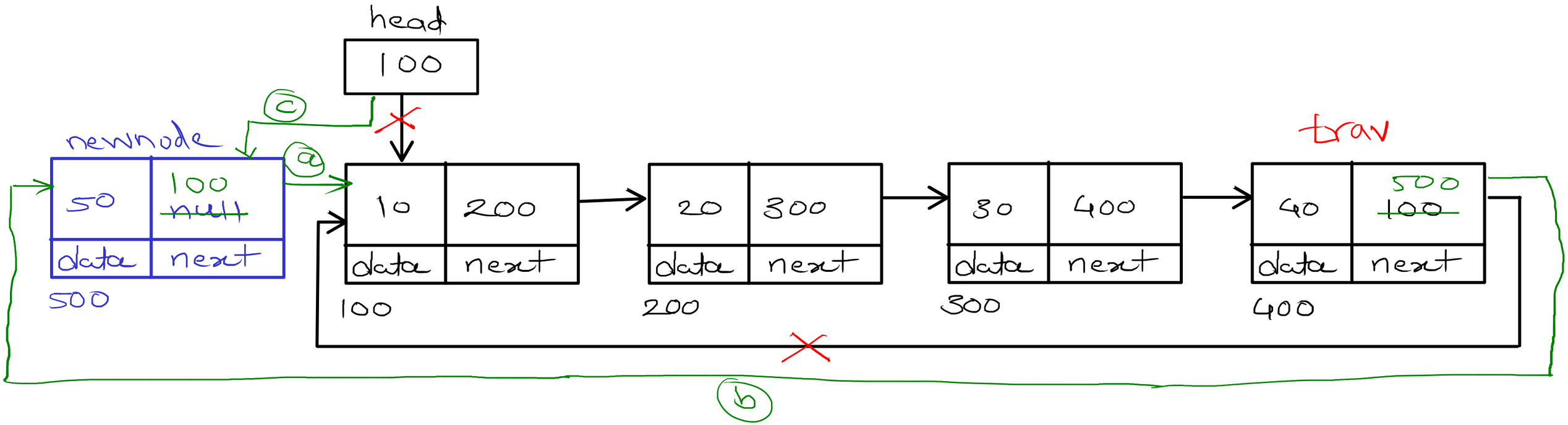


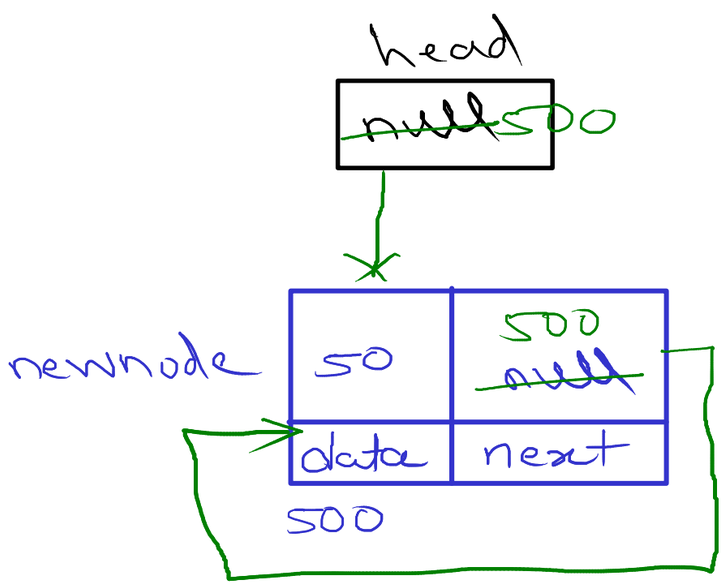
//1. start at head
//2. print current node
//3. go on next node
//4. repeat step 2 and 3 till last node

```
trav = head;
do {
    sysout(trav.data);
    trav = trav.next;
} while(trav != head)
```

Time Complexity : O(n)

Singly Circular Linked List - Add First

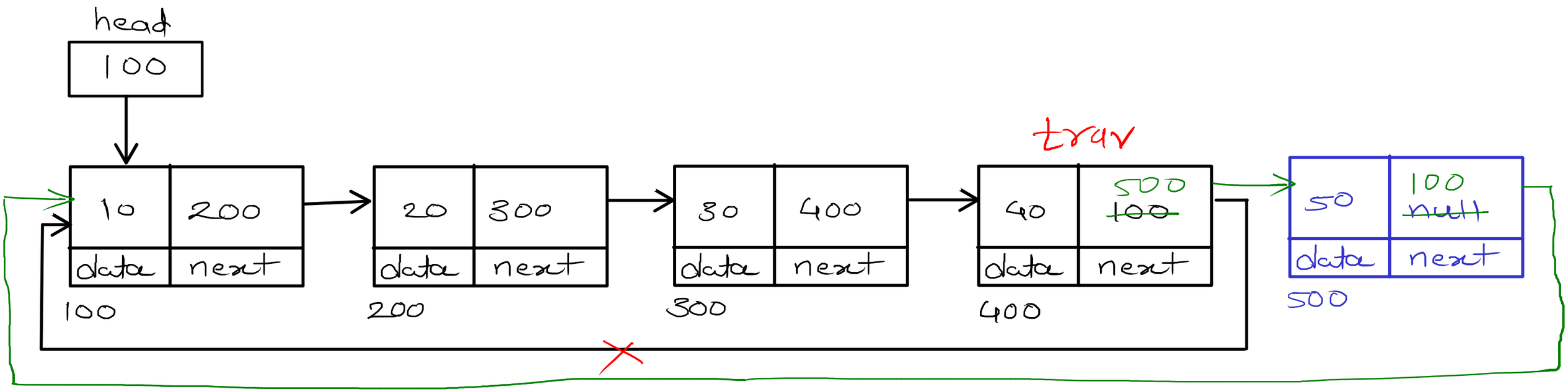




Time Complexity : O(n)

```
//1. create node with given data
//2. if list is empty
      //a. add newnode into head
      //b. make list circular
//3. if list is not empty
      //a. add first node into next of newnode
      // traverse till last node
      //b. add newnode into next of last node
      //c. move head on newnode
```

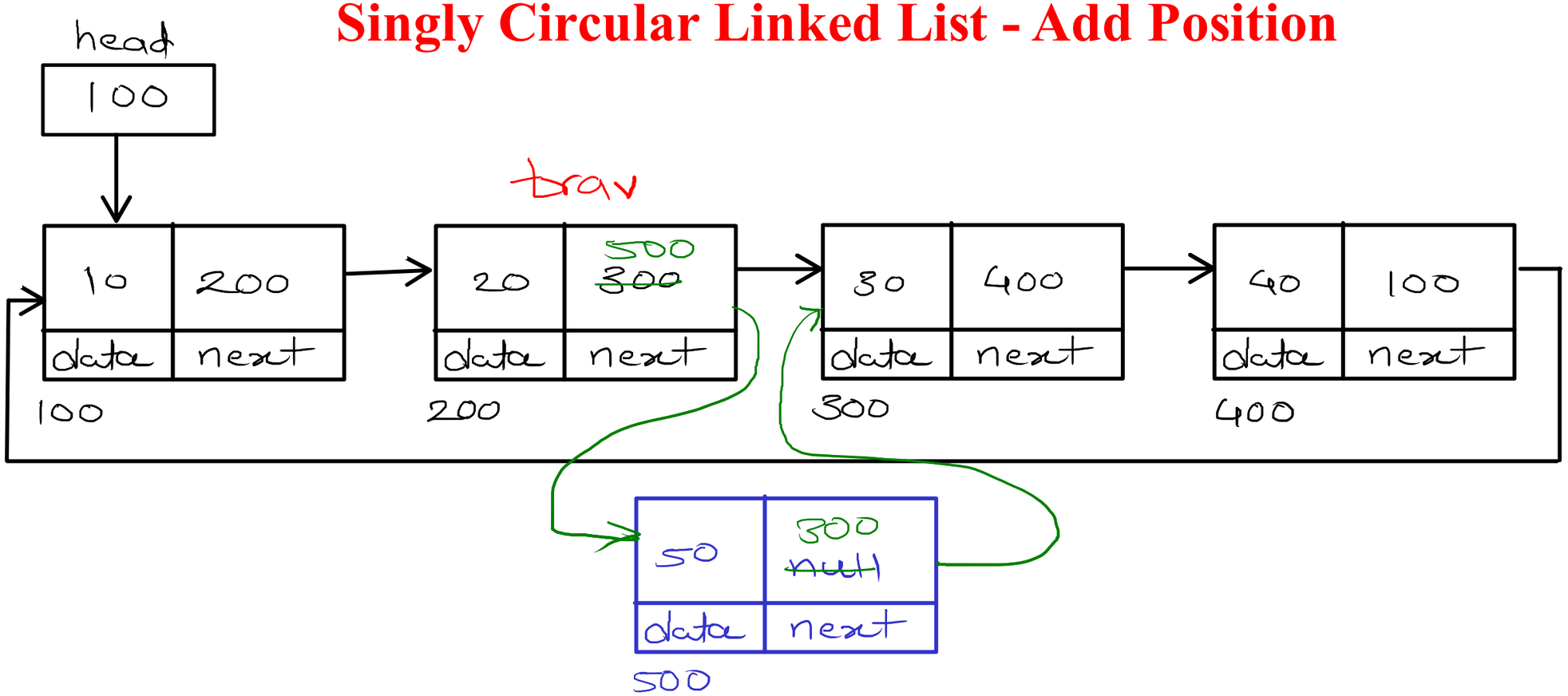
Singly Circular Linked List - Add Last




```
//1. create node
//2. if list is empty
        //a. add newnode into head
        //b. make list circular
//3. if list is not empty
        //a. first node into next of newnode
        //b. traverse till last node
        //c. add newnode into next of last node
```

Time Complexity : O(n)

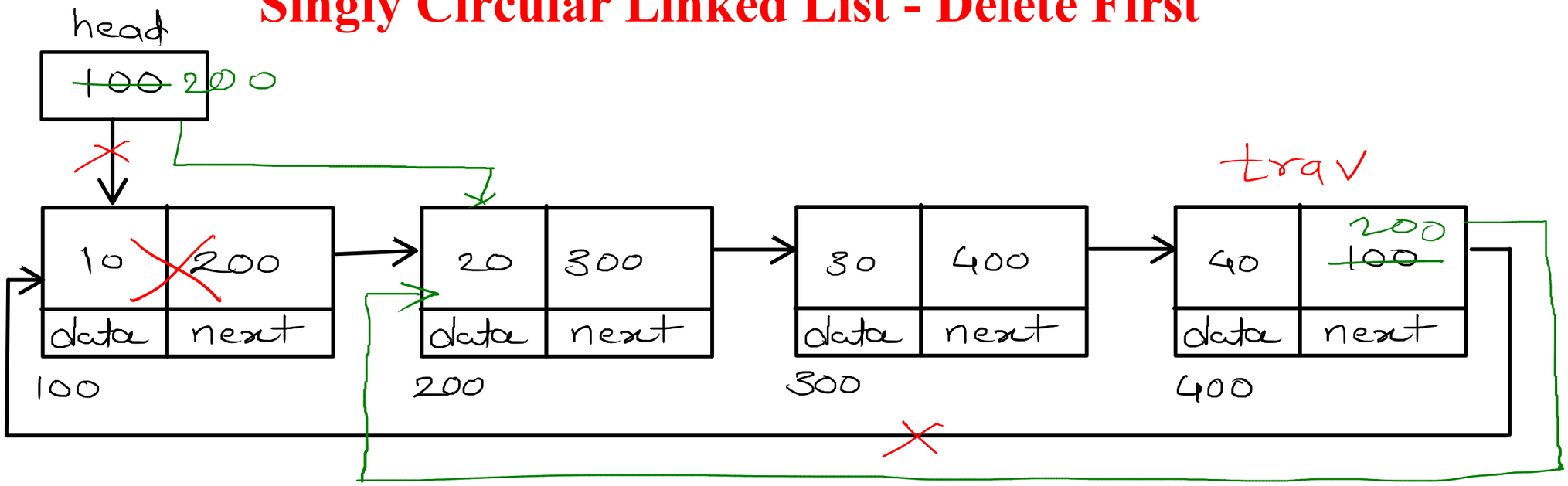
Singly Circular Linked List - Add Position

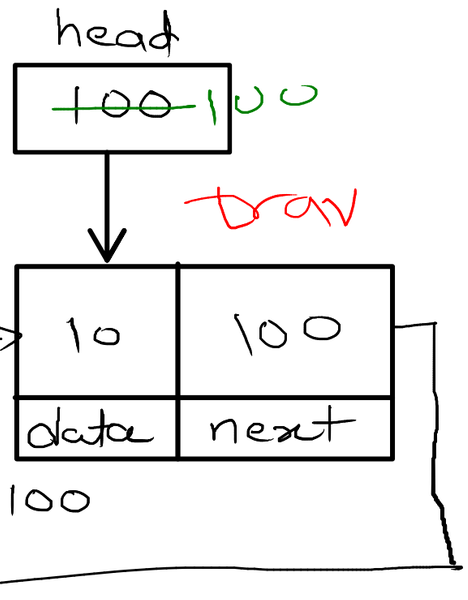


Time Complexity : O(n)

```
//1. create node
//2. if list is empty
        //a. add newnode into head
        //b. make it circular
//3. if list is not empty
        //a. traverse till pos-1 node
        //b. add pos node into next of newnode
        //c. add newnode into next of pos-1 node
```

Singly Circular Linked List - Delete First



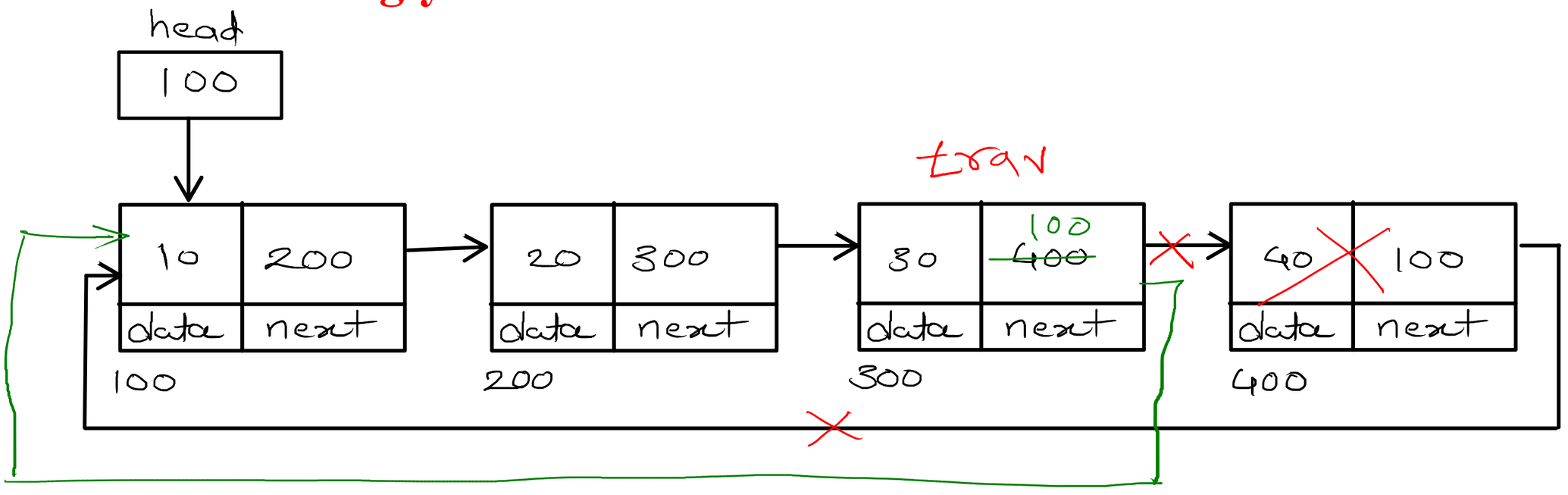



```
head = head.next;
trav = head;
while(trav.next != head)
    trav = trav.next;
trav.next = head;
```

```
//1. if list is empty
      return;
//2. if list has single node
      // add null into head
//3. if list has multiple nodes
      //a. traverse till last node
      //b. move head on second node
      //c. add second node into next of last node
```

Time Complexity : O(n)

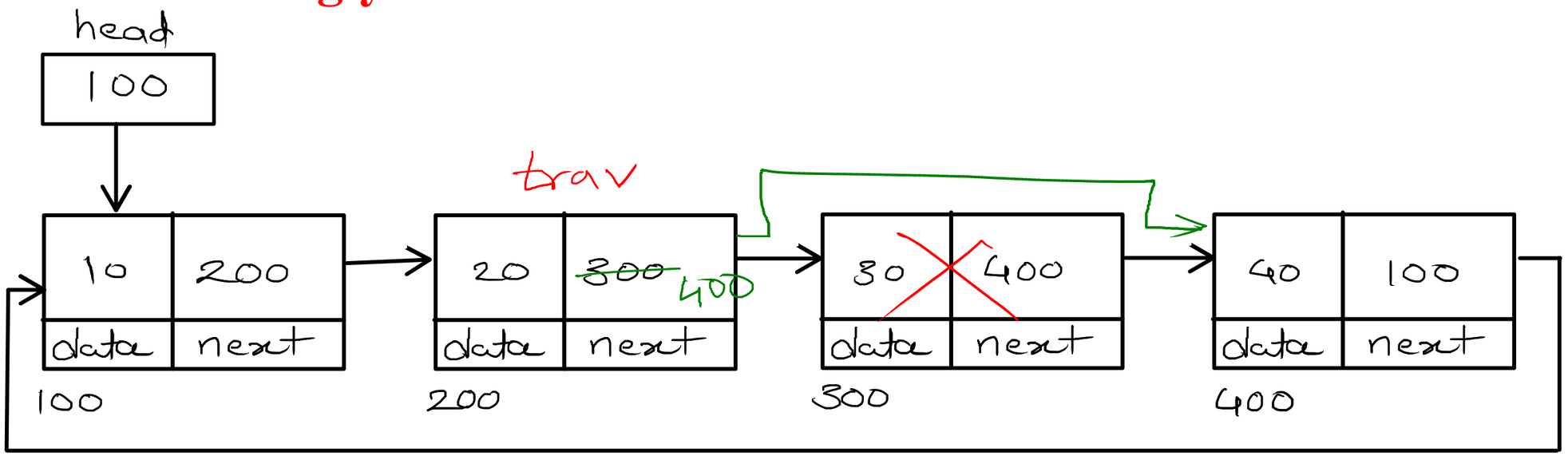
Singly Circular Linked List - Delete Last




```
//1. if list is empty
        return;
//2. if list has single node
        // add null into head
//3. if list has multiple nodes
        //a. traverse till second last node
        //b. add head into next of second last node
```

Time Complexity : O(n)

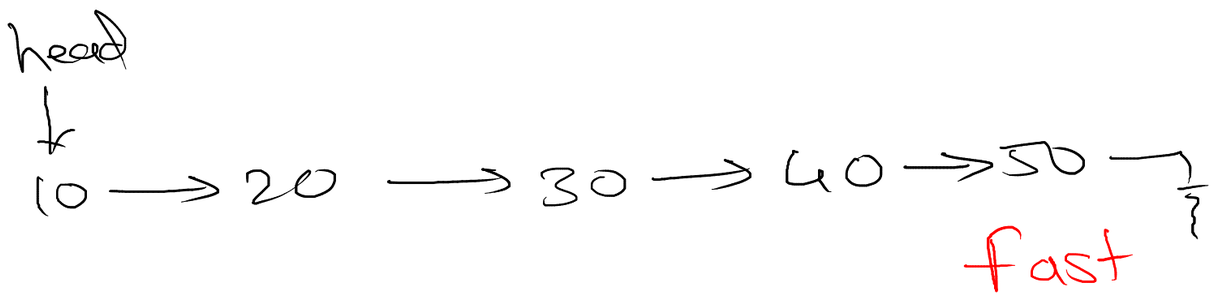
Singly Circular Linked List - Delete Position

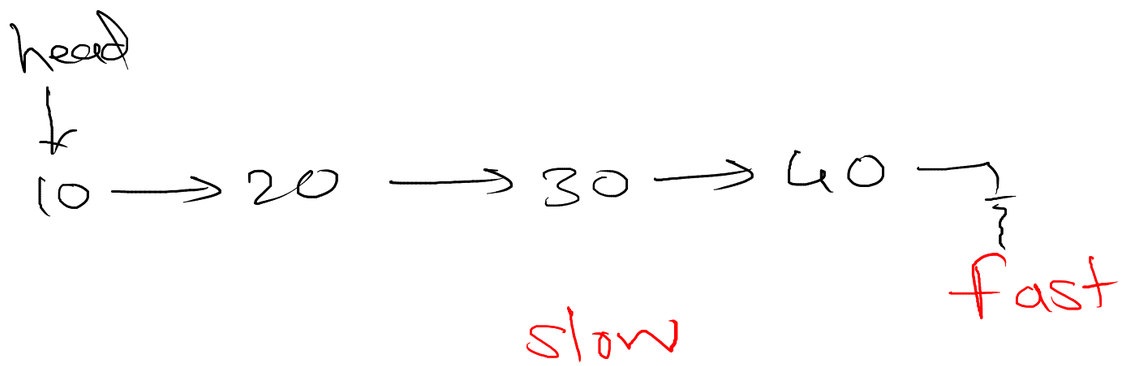



```
//1. if list is empty
        return;
//2. if list has single node
        // add null into head
//3. if list has multiple nodes
        //a. traverse till pos-1 node
        //b. add pos+1 node into next of pos-1 node
```

Time Complexity : O(n)

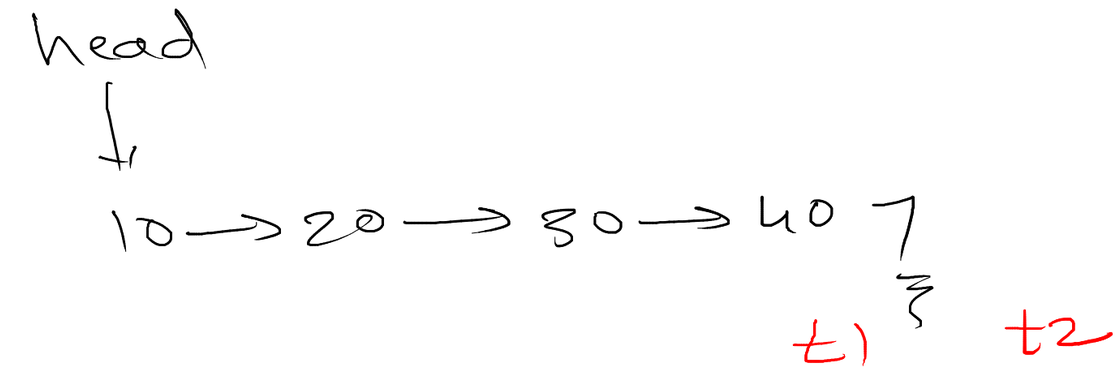
Find Mid



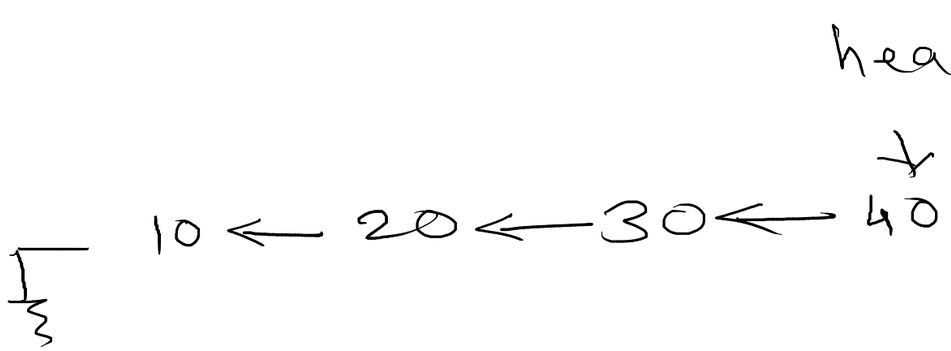



```
fast = head;
slow = head;
while(fast != null && fast.next != null)
{
    fast= fast.next.next;
    slow =slow.next;
}
sysout(slow.data);
```

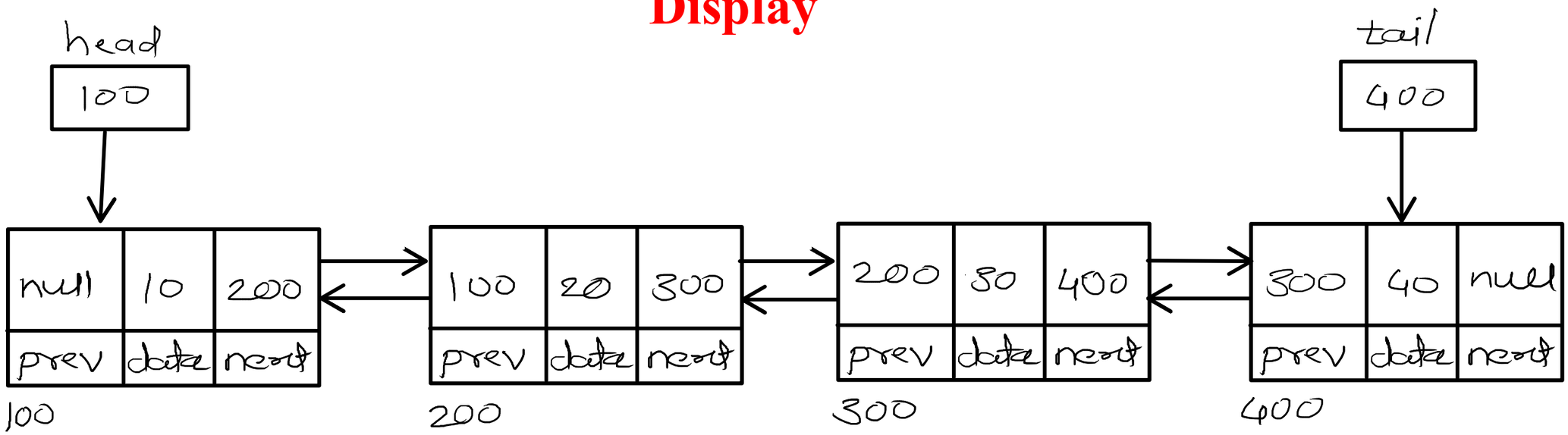
Reverse Linked List




```
t1 = head;
t2 = t1.next;
while(t2 != null ) {
    t3 = t2.next;
    t2.next = t1;
    t1 = t2;
    t2 = t3;
}
head.next = null;
head = t1;
```



Doubly Linear Linked List
Display



Forward Traversal

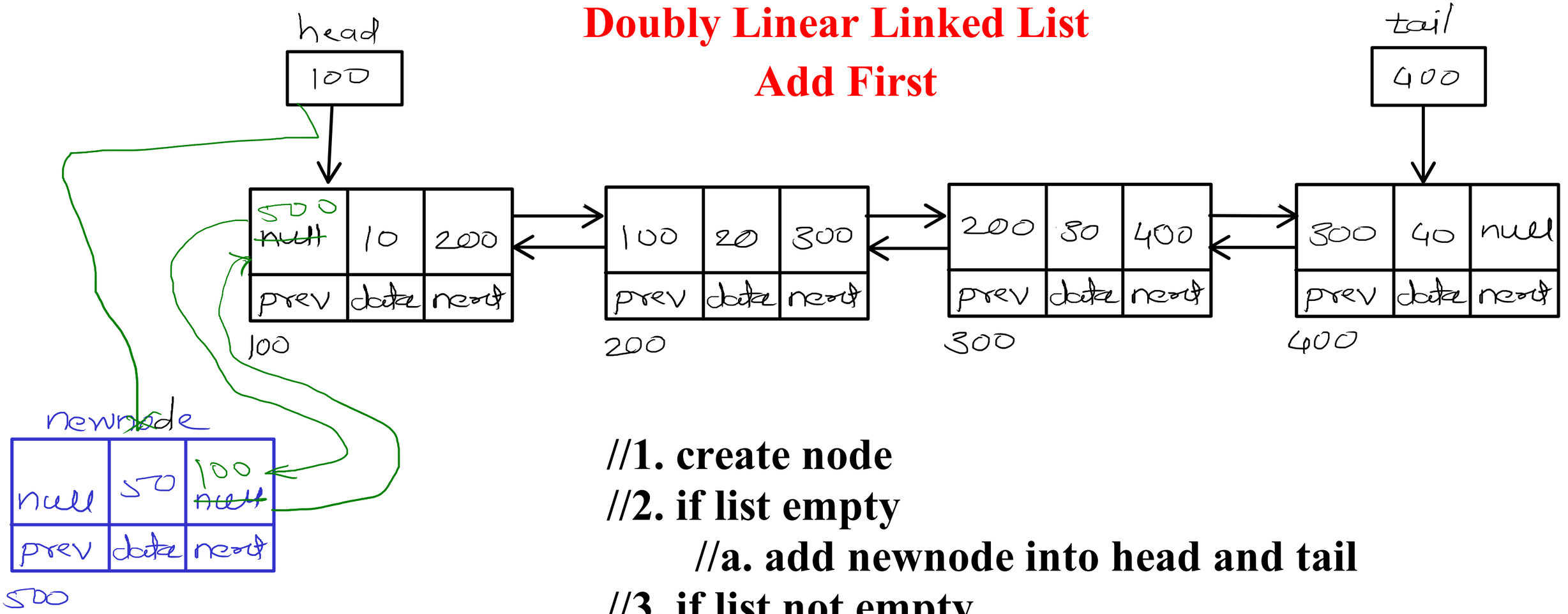
//1. create trav and start at head
//2. print data of current node
//3. go on next node
//4. repeat step 2 and 3 till last node

Reverse Traversal

//1. create trav and start at tail
//2. print data of current node
//3. go on prev node
//4. repeat step 2 and 3 till first node

Time Complexity : O(n)

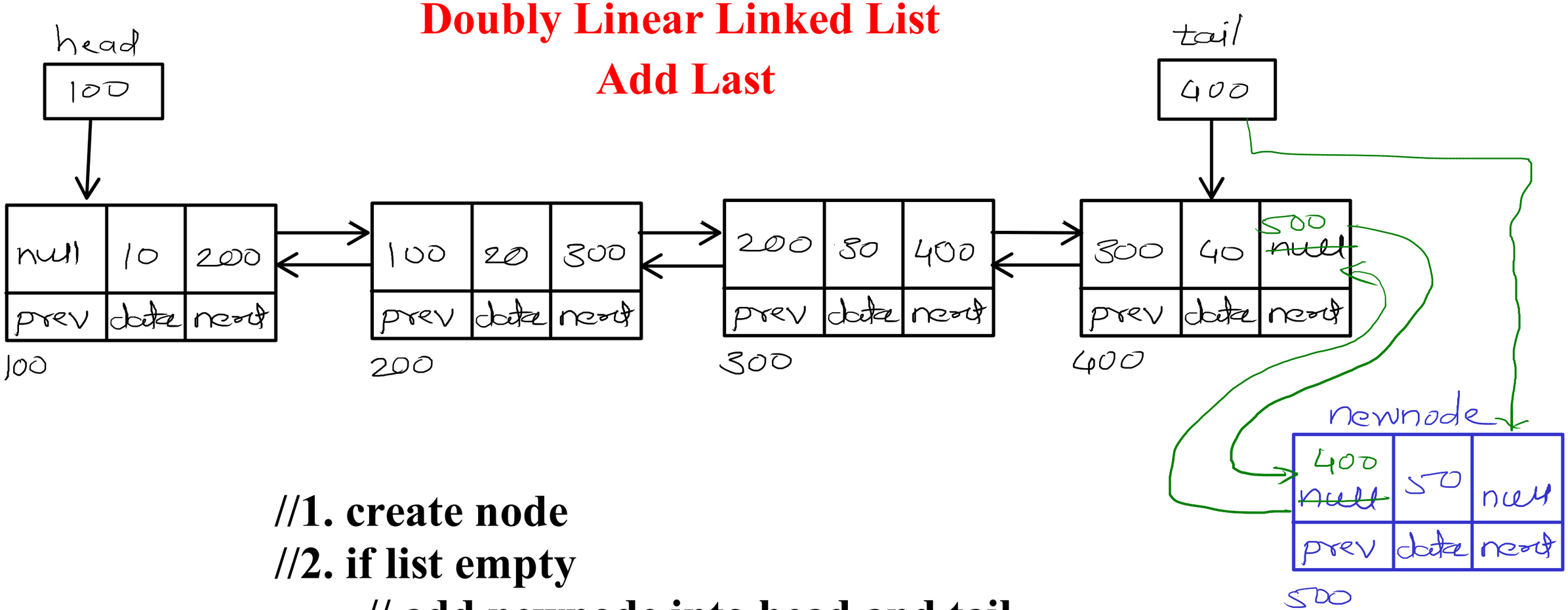
Doubly Linear Linked List
Add First



//1. create node
//2. if list empty
//a. add newnode into head and tail
//3. if list not empty
//a. add first node into next of newnode
//b. add newnode into prev of first node
//c. move head on newnode

Time Complexity : O(1)

Doubly Linear Linked List
Add Last



//1. create node
//2. if list empty
// add newnode into head and tail
//3. if list is not empty
//a. add last node into prev of newnode
//b. add newnode into next of last node
//c. move tail on newnode

Time Complexity : O(1)