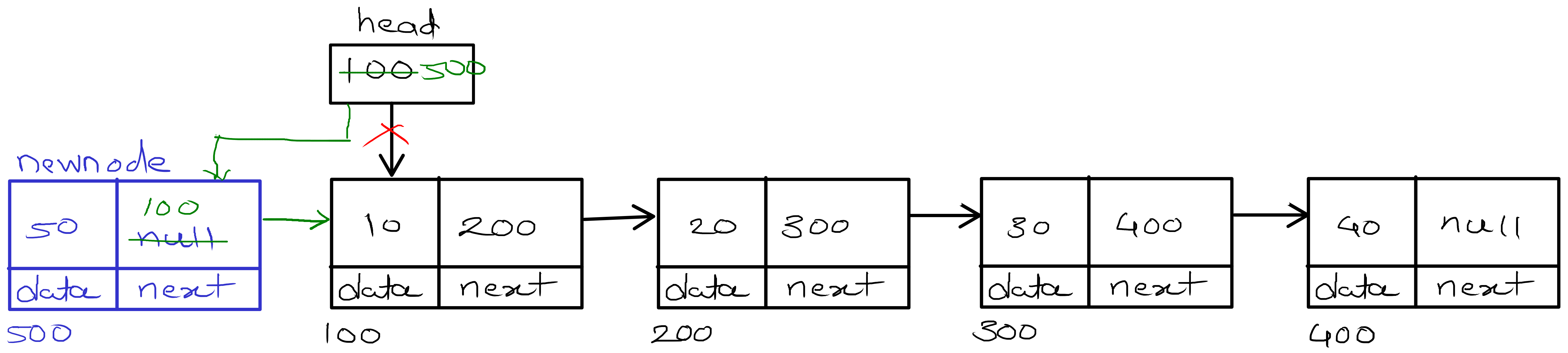
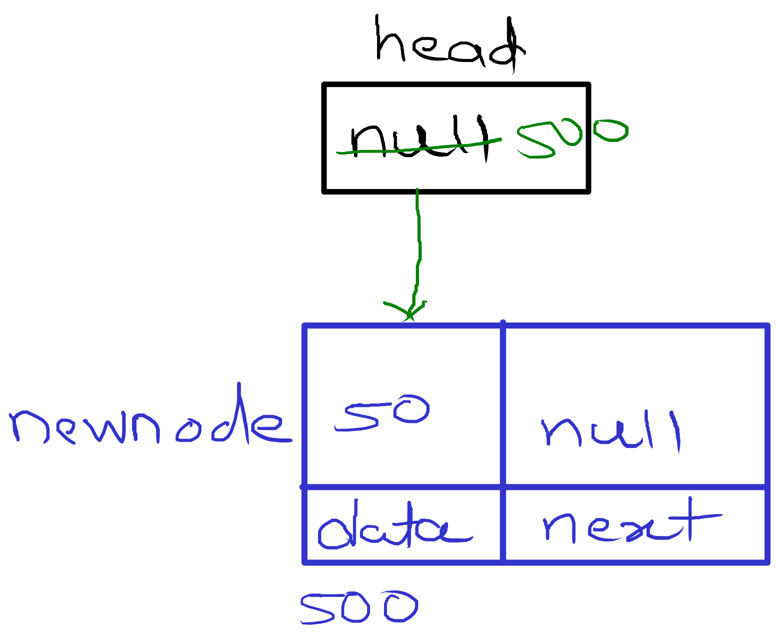
Singly Linear Linked List - Add First

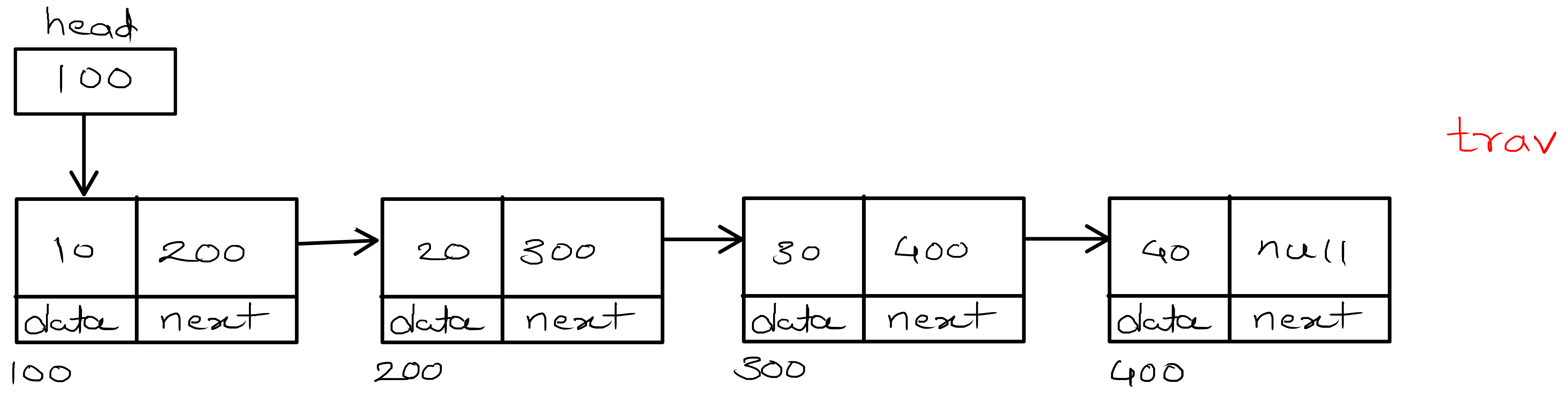




```
//1. create node with given value
//2. if list is empty
        //a. add newnode into head itself
//3. if list is not empty
        //a. add first node into next of newnode
        //b. add newnode into head
```

Time Complexity : O(1)

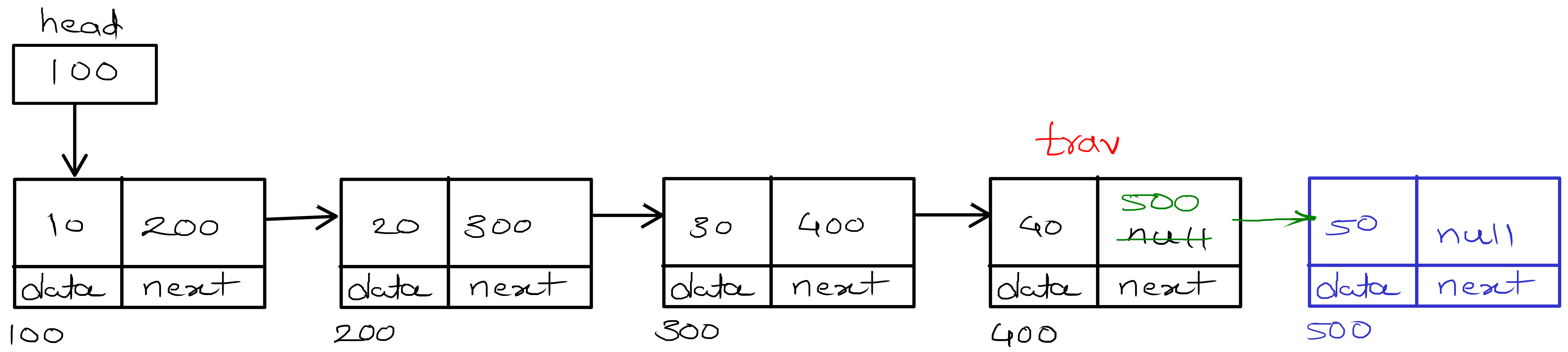
Singly Linear Linked List - Display



//1. create one referance and start at first node
//2. print(visit) the current node
//3. go on next node
//4. repeat step 2 and 3 till last node

Time Complexity : O(n)

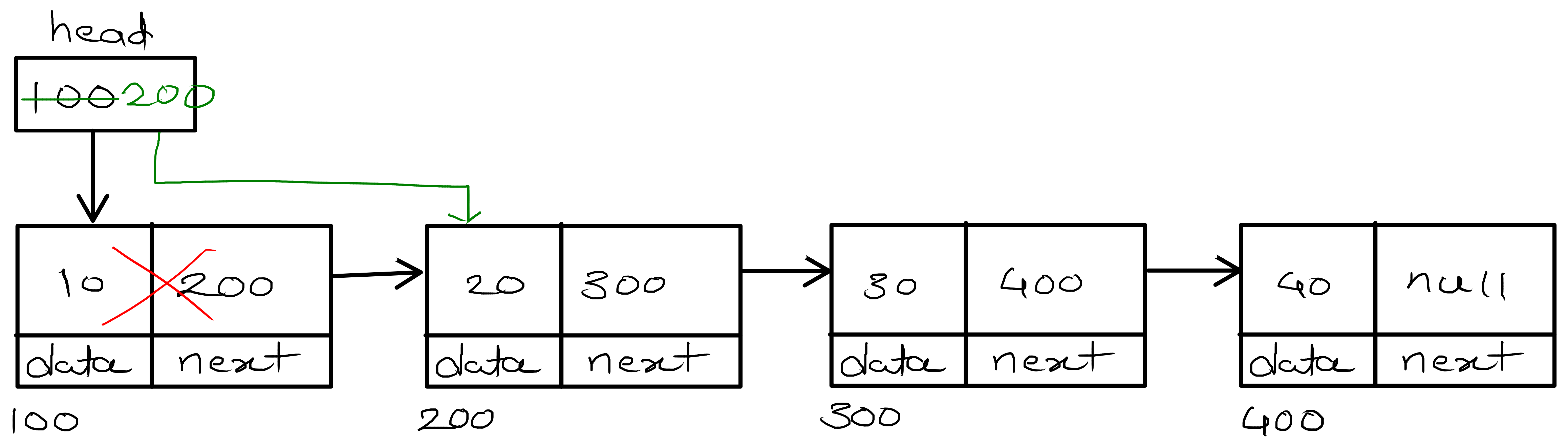
Singly Linear Linked List - Add Last

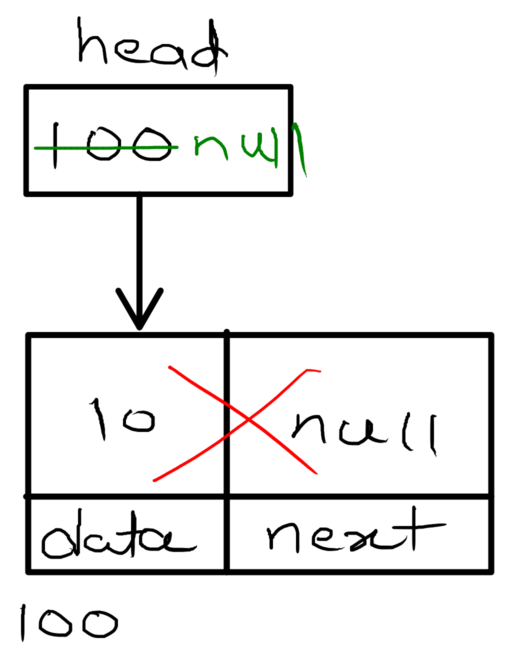


```
//1. create node with given data
//2. if list is empty
        //a. add newnode into head itself
//3. if list is not empty
        //a. traverse till last node
        //b. add newnode into next of last node(trav.next)
```

Time Complexity : O(n)

Singly Linear Linked List - Delete First

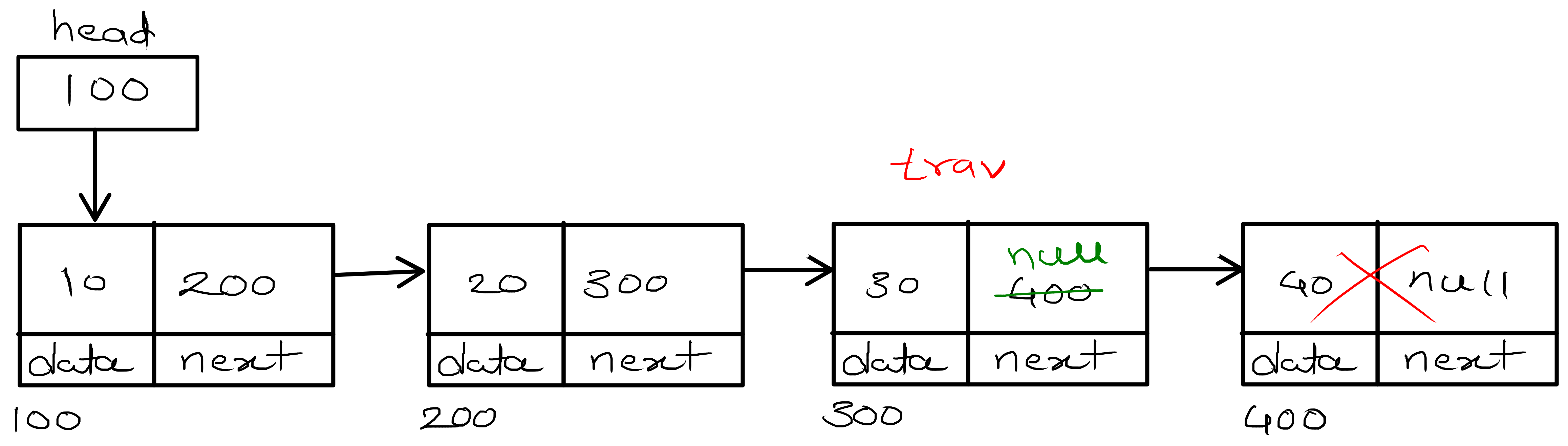


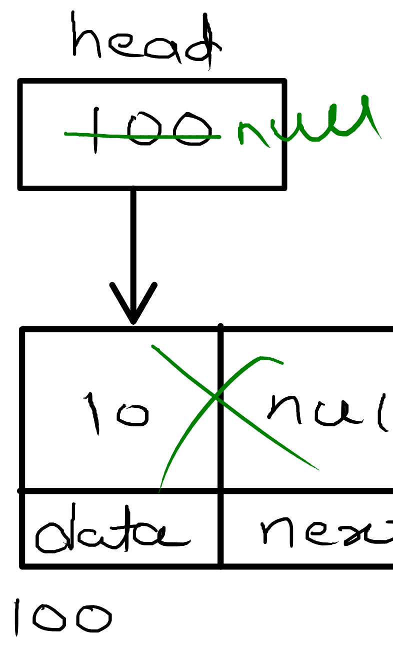


```
//1. if list is empty
        // print msg
//2. if list is not empty
        //a. move head on second node
```

Time Complexity : O(1)

Singly Linear Linked List - Delete Last

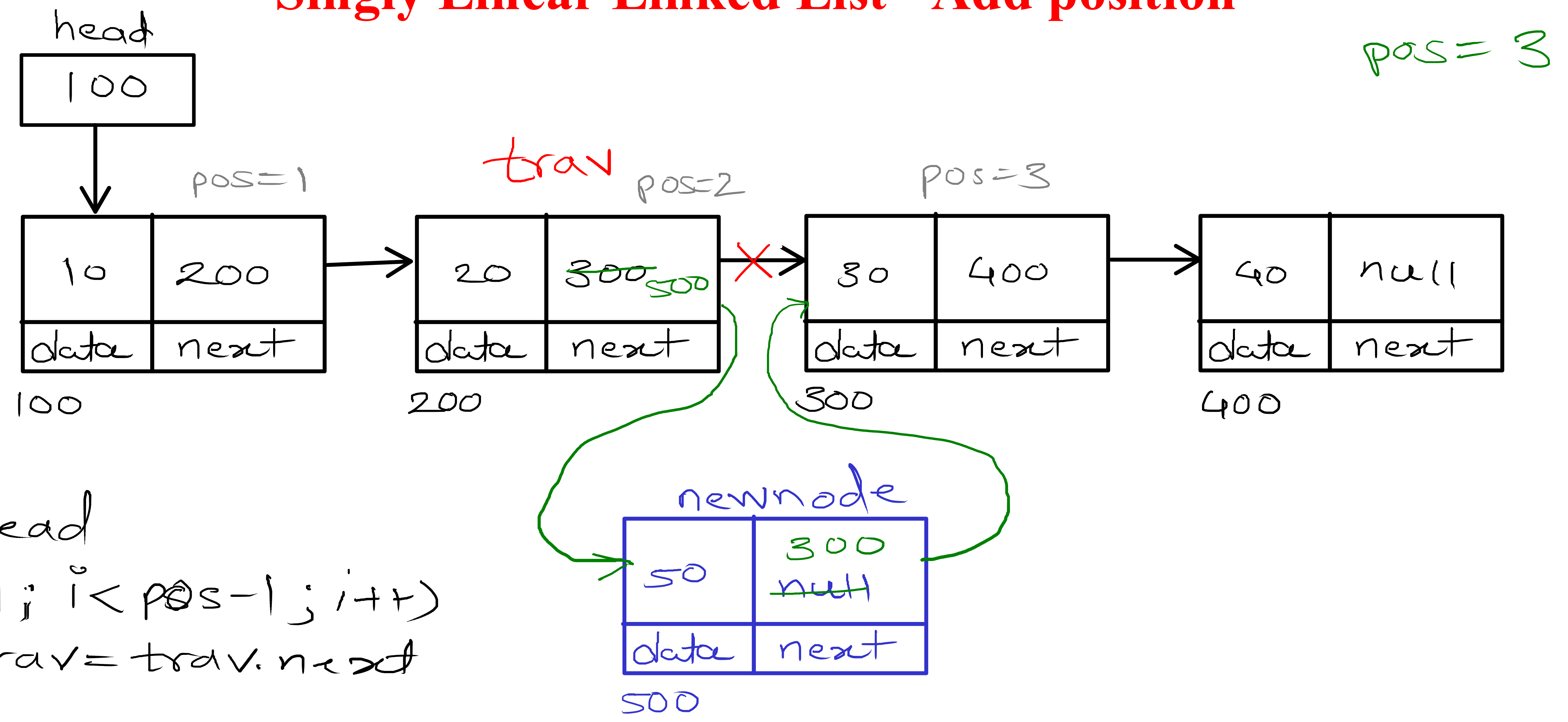




```
//1. if list is empty
        // print msg
//2. if list has single node
        // make head equal to null
//2. if list has multiple nodes
        //a. taverse till second last node
        //b. make next of second last node equal to null
```

Timp complexity : O(n)

Singly Linear Linked List - Add position

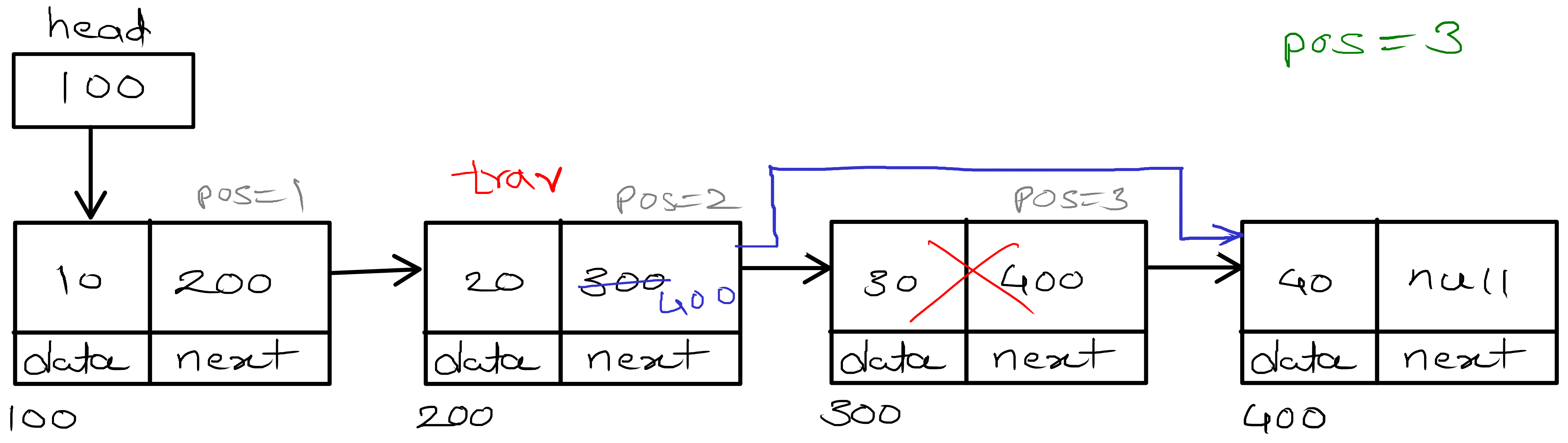


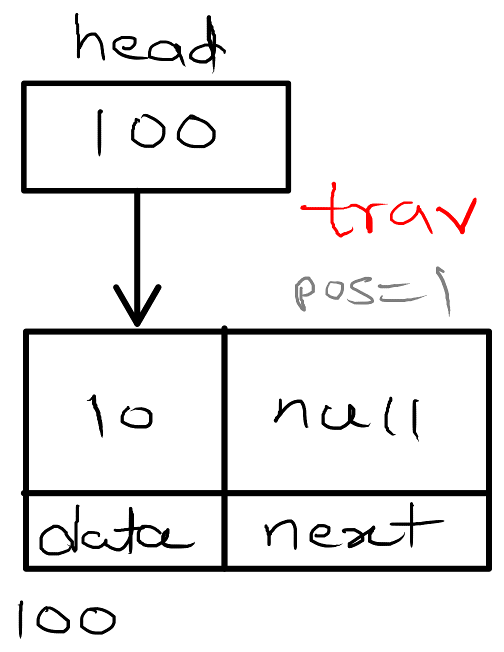
```
trav=head
for(i=1; i<pos-1; i++)
        trav=trav.next
```

//1. create node
//2. if list is empty
    //a. add newnode into head
//3. if list is not empty
    //a. traverse till pos-1
    //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





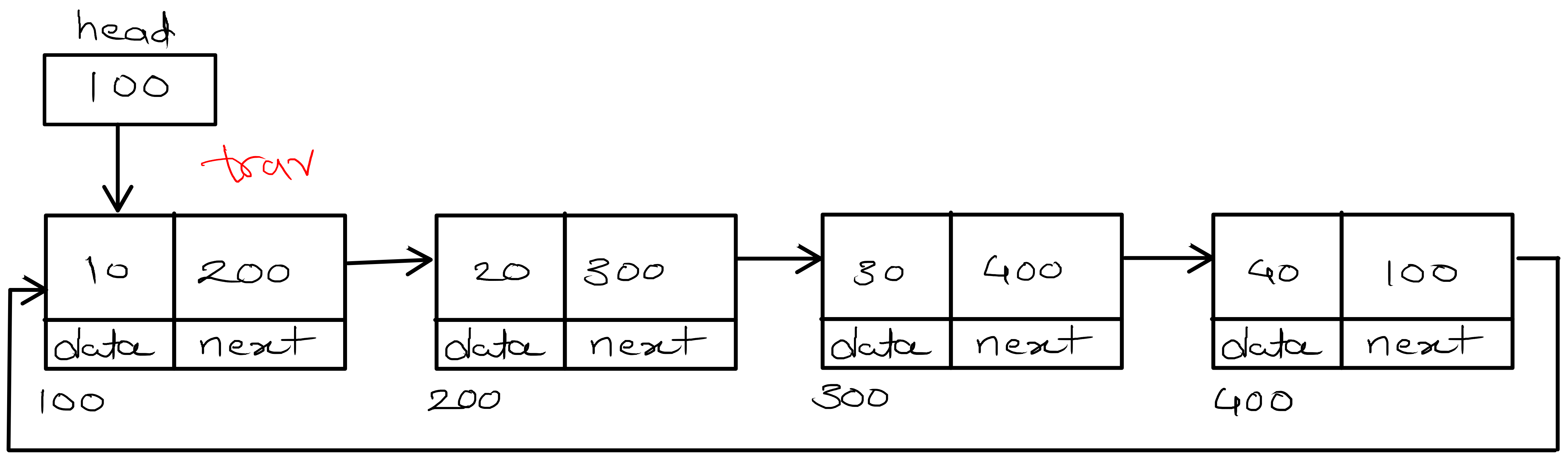
```
//1. if list is empty
        // print msg
//2. if list is not empty
        //a. traverse till pos -1 node
        //b. add pos+1 node into next of pos-1 node
```

Time Complexity : O(n)

```
else {
    trav = head;
    for(i=1; i< pos-1 && trav.next.next != null; i++)
                trav = trav.next;
  ->trav.next = trav.next.next;
                    null.
}
```

deletePosition(1)
deletePosition(2)

Singly Circular Linked List - Display

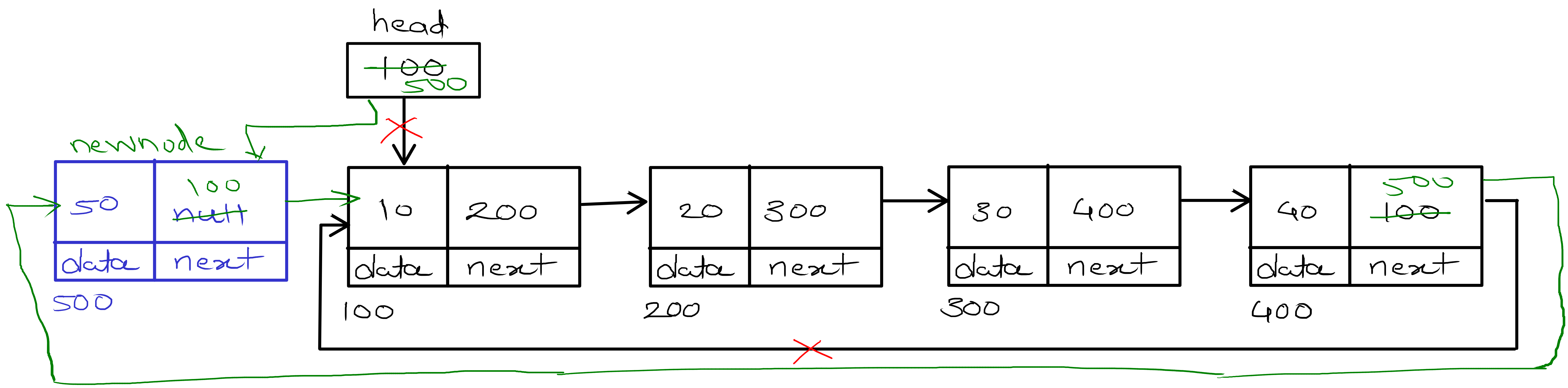


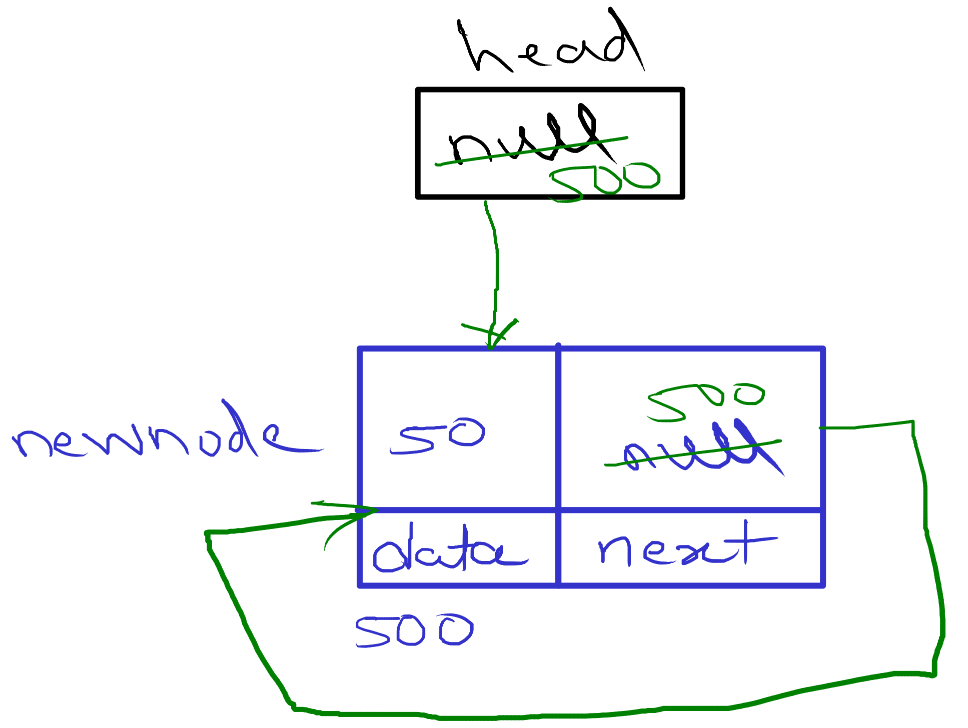
//1. create trav and start at head
//2. visit(print) current node
//3. go on next node
//4. repeat step 2 and 3 till last node

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

Time Complexity : O(n)

Singly Circular Linked List - Add First

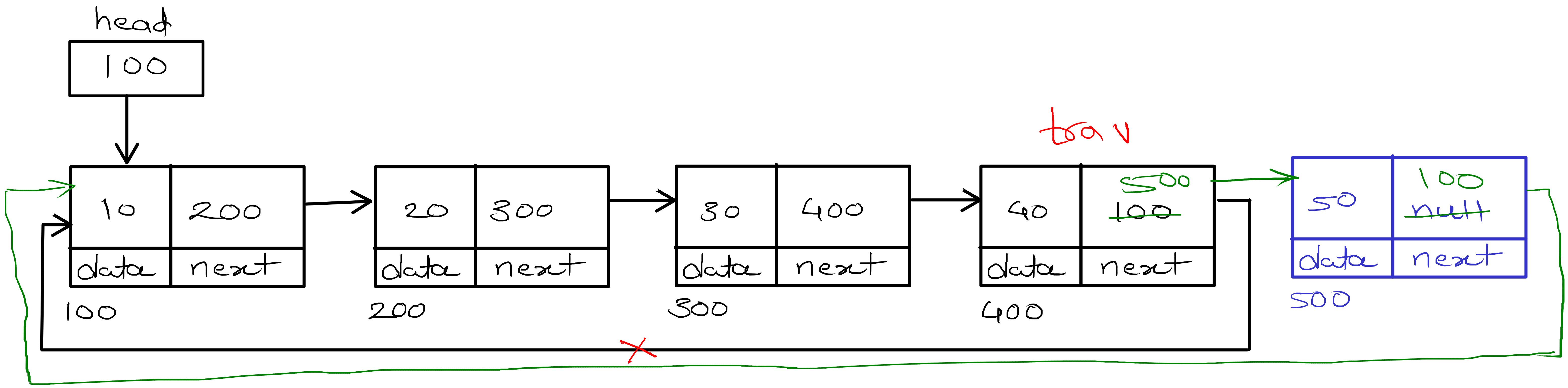




```
//1. create node with given data
//2. if list is empty
        //a. add newnode into head
        //b. make list circular
//3. if list is not empty
        //a. add first node into next of newnode
        //b. traverse till last node
        //c. add newnode into next of last node
        //d. move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

Time Complexity : O(n)

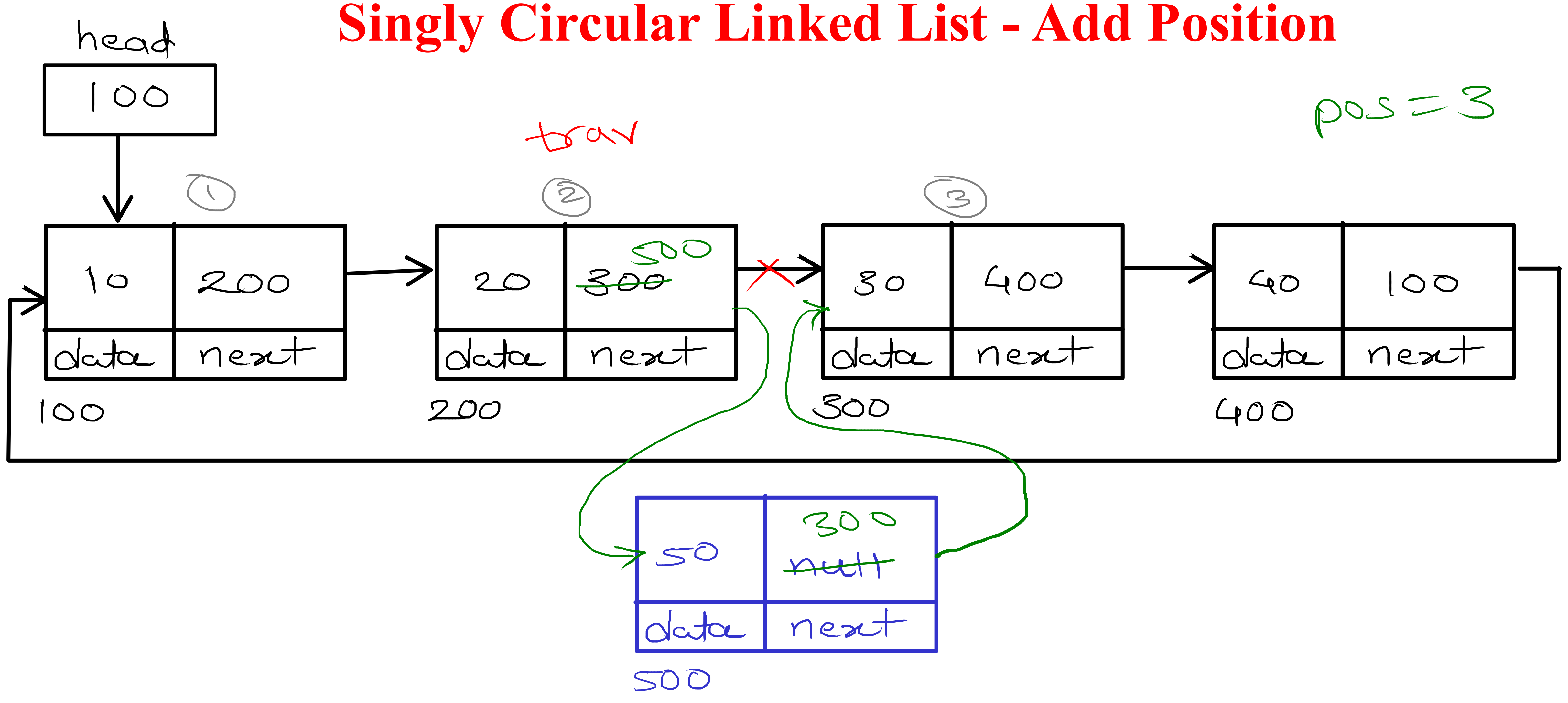
Singly Circular Linked List - Add Last



```
//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
        //b. traverse till last node
        //c. add newnode into next of last node
```

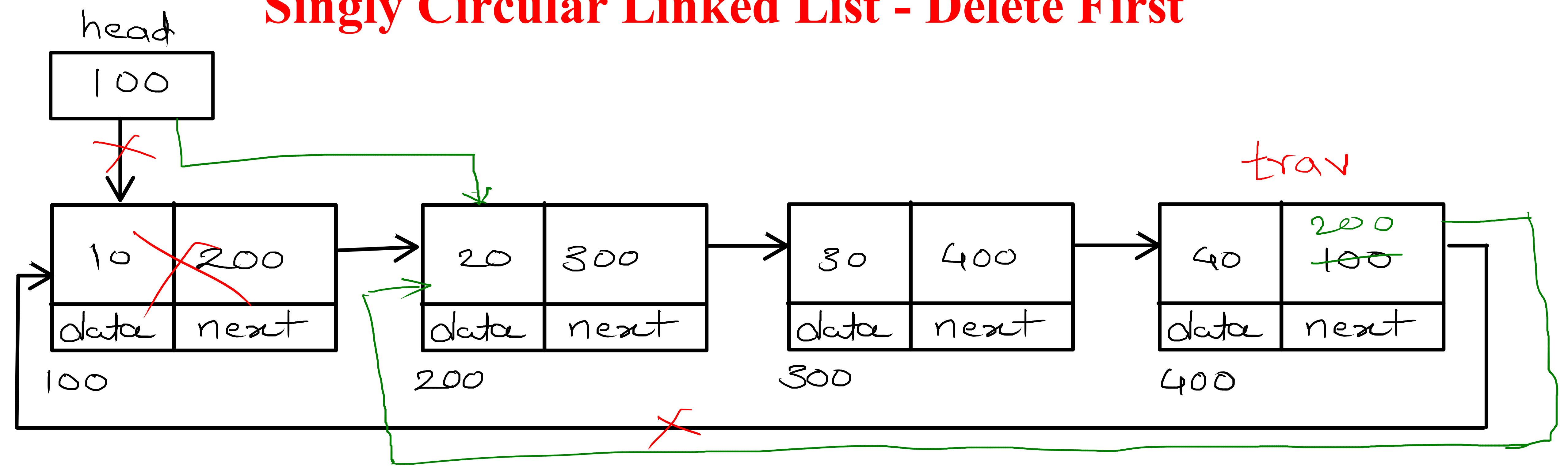
Time Complexity : O(n)

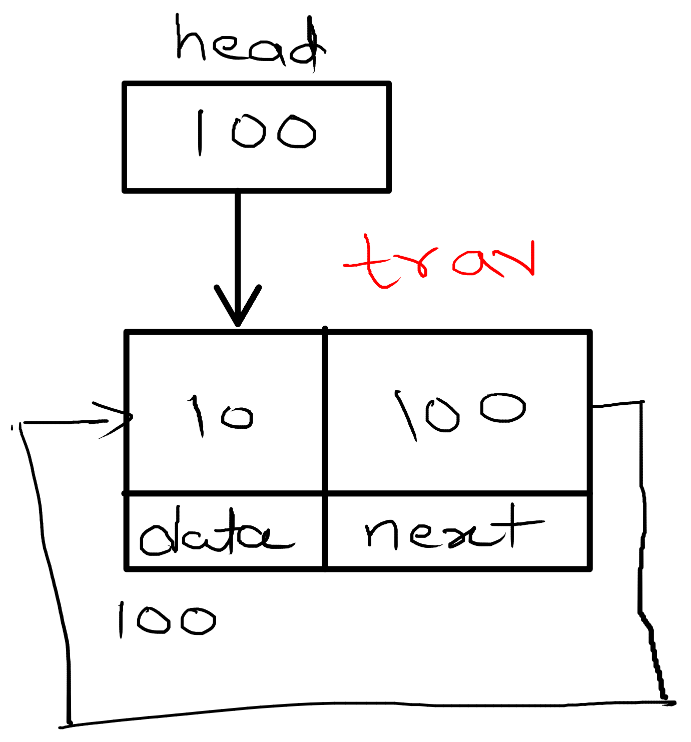
Singly Circular Linked List - Add Position



```
//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
    //b. add pos node into next of newnode
    //c. add newnode into next of pos-1 node
```

Singly Circular Linked List - Delete First

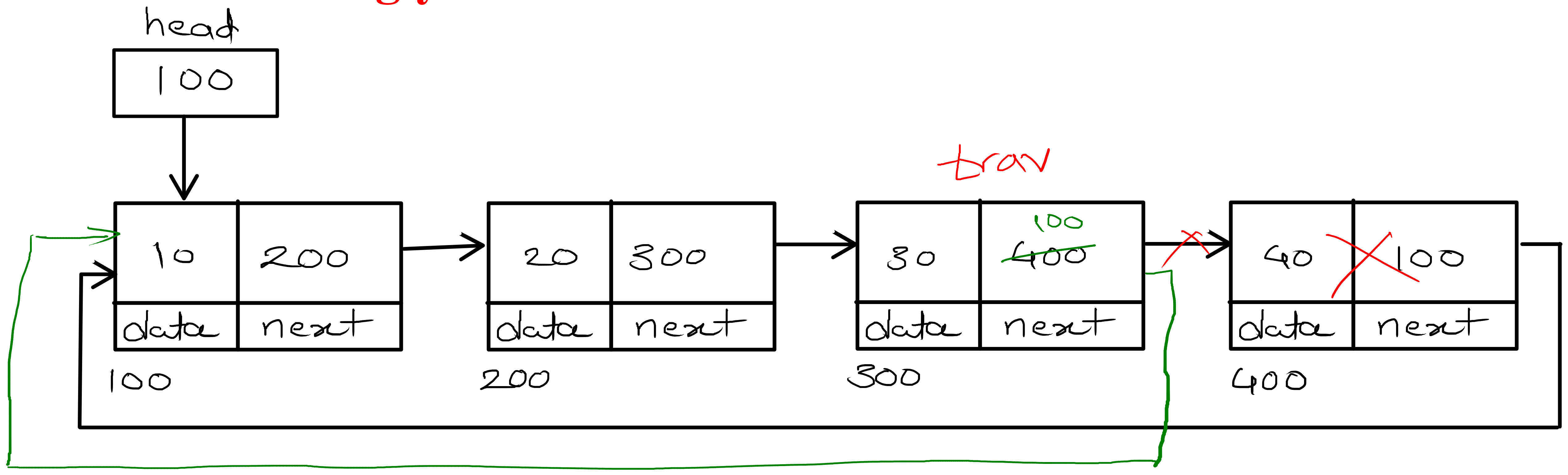




```
//1. if list is empty
        // print msg
//2. if list has single node
        // make head = null
//3. if list has multiple nodes
        //a. traverse till last node
        //b. add second node into next of last node
        //c. move head on second node
```

Time complexity : O(n)

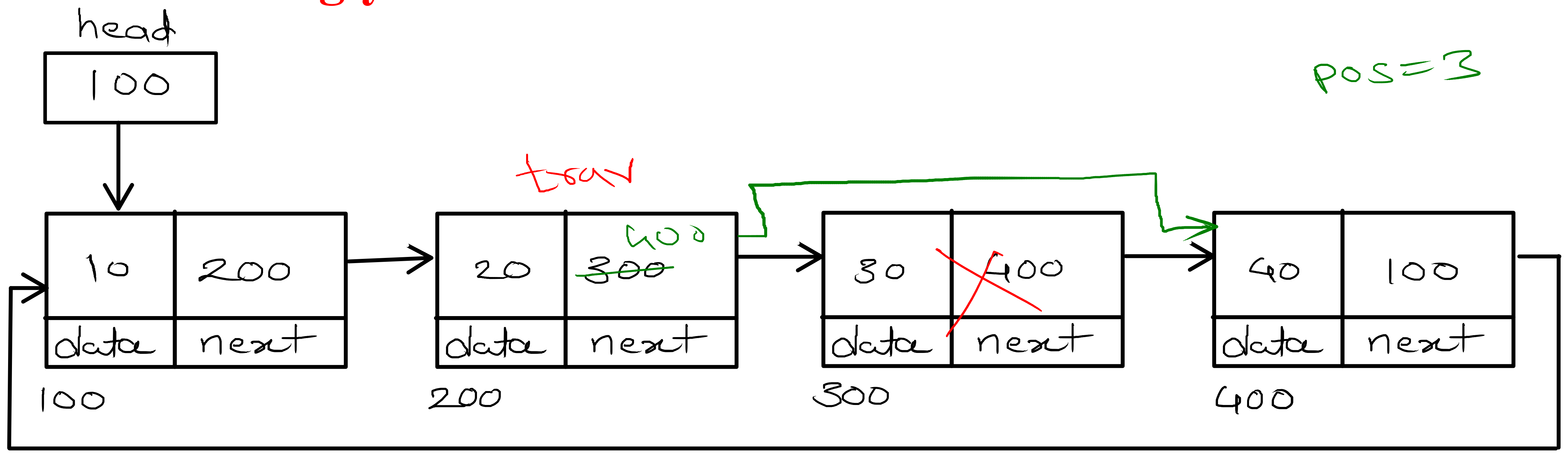
Singly Circular Linked List - Delete Last



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

Time complexity : O(n)

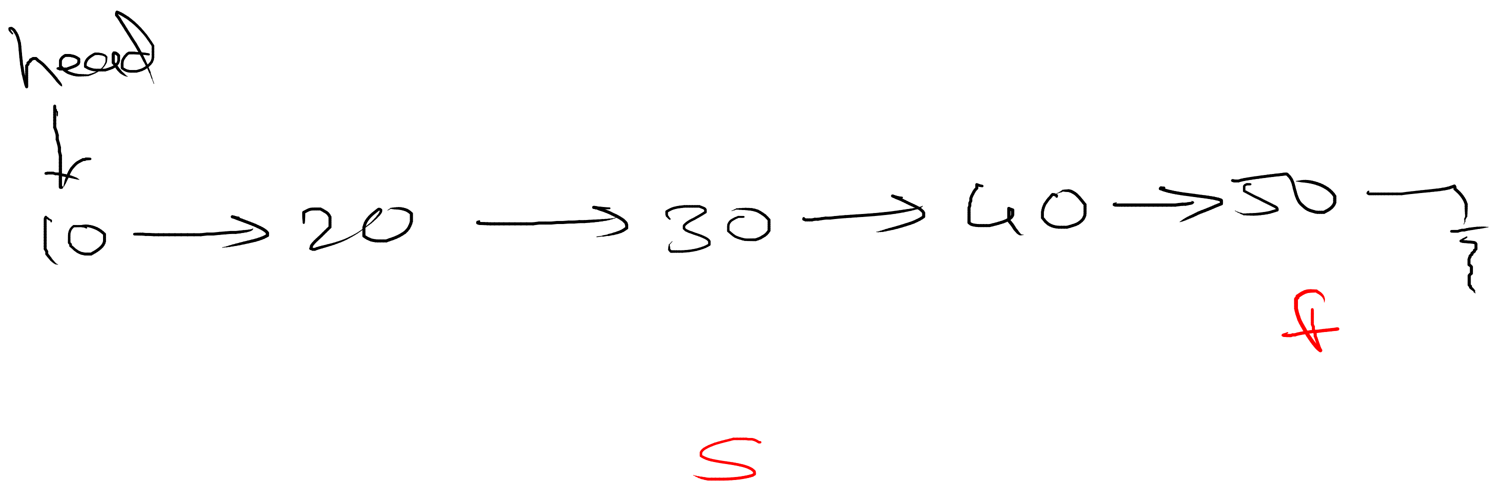
Singly Circular Linked List - Delete Position



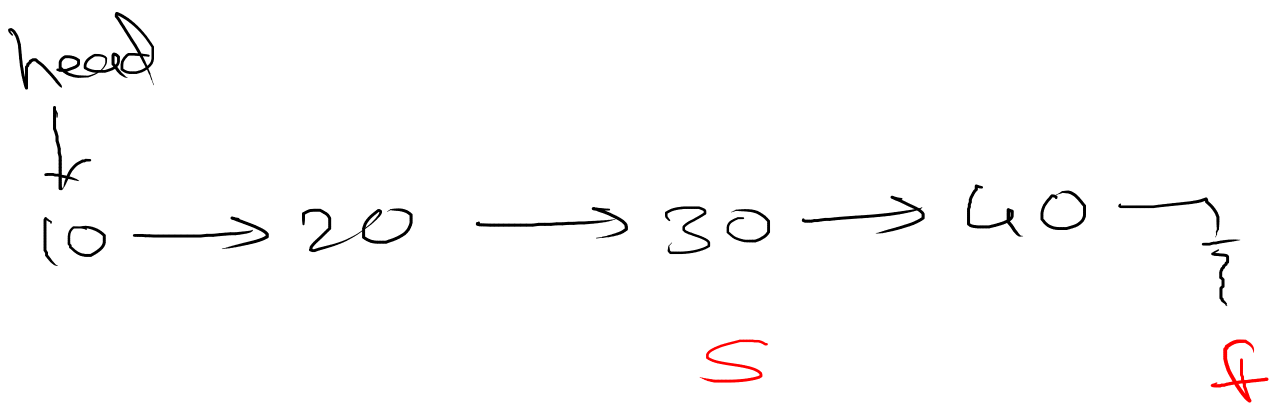
```
//1. if list is empty
        // print msg and return
//2. if list has single node
        // make head = null
//3. if last has multiple node
        //a. traverse till pos - 1 node
        //b. add pos+1 node into next of pos-1 node
```

Time complexity : O(n)

Find Mid



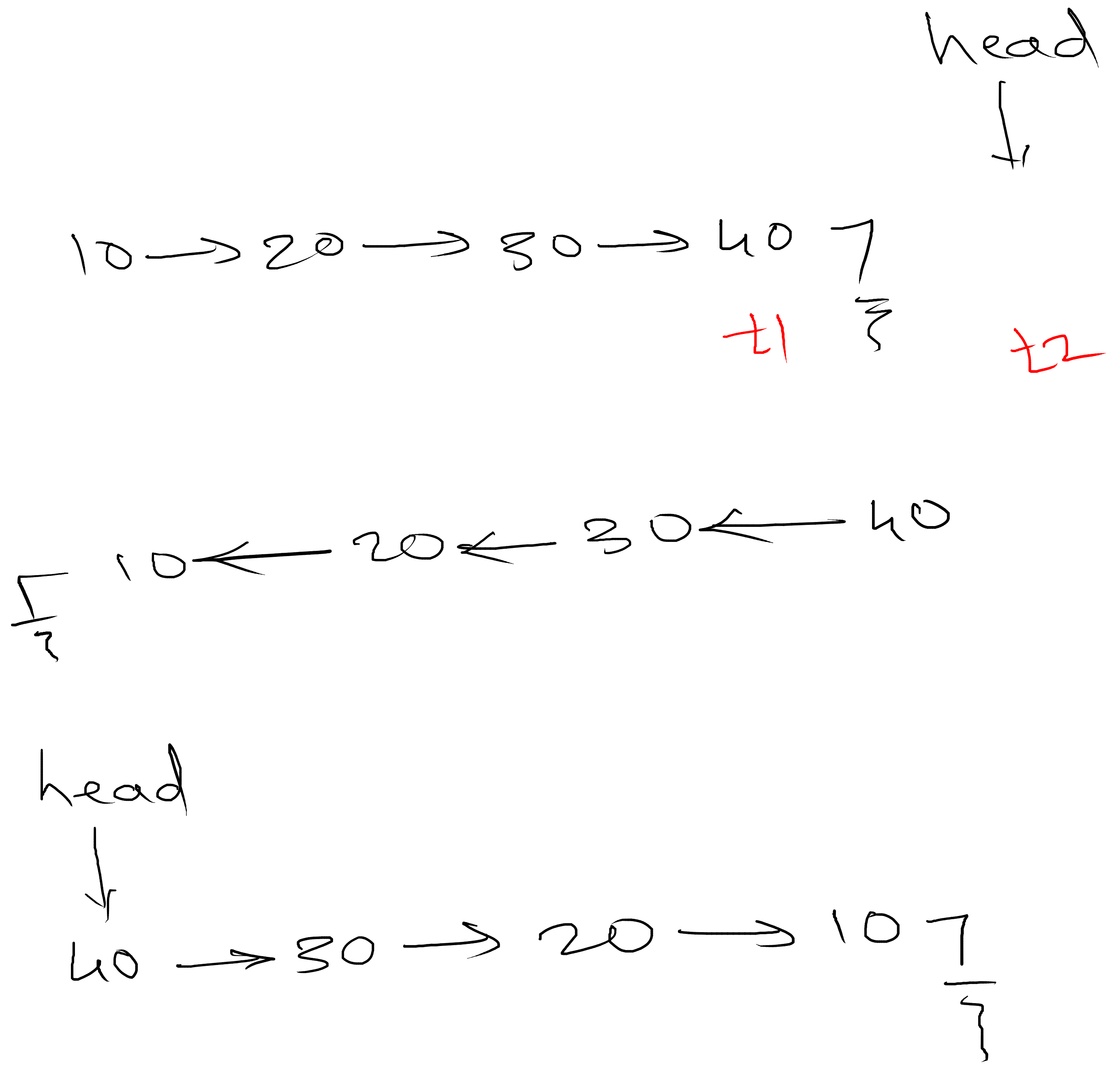
```
f = head
s = head
while (f.next != null)
{  f = f.next.next;
   s = s.next;
}
```



Time complexity : O(n)

```
f = head
s = head
while ( f != null && f.next != null)
{  f = f.next.next;
   s = s.next;
}
```

Reverse Linked List



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

Time complexity : O(n)