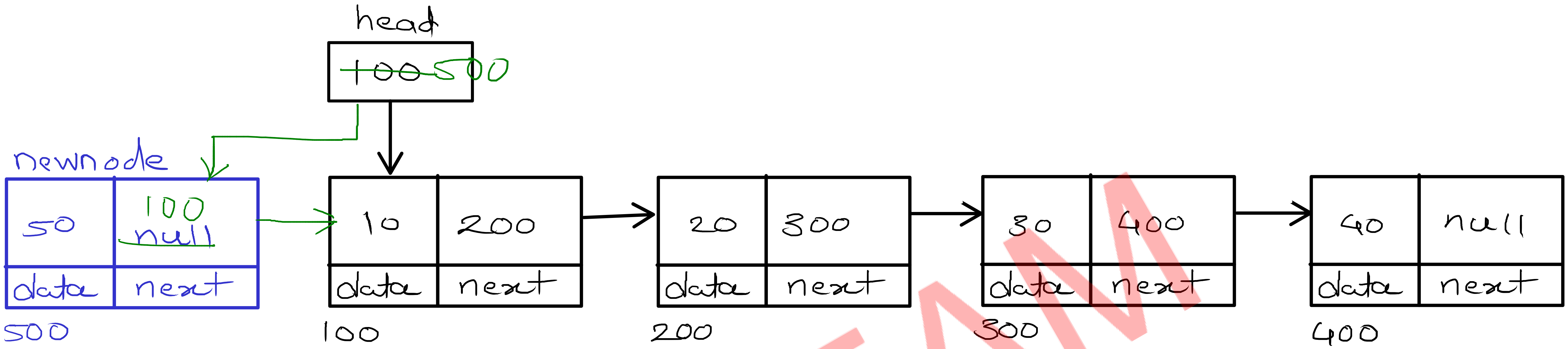
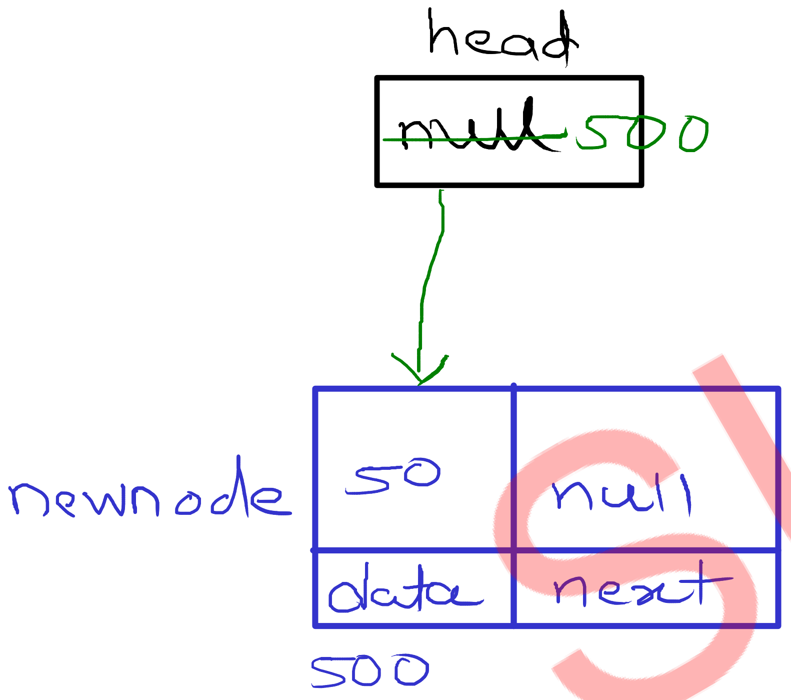
# Singly Linear Linked List - Add First

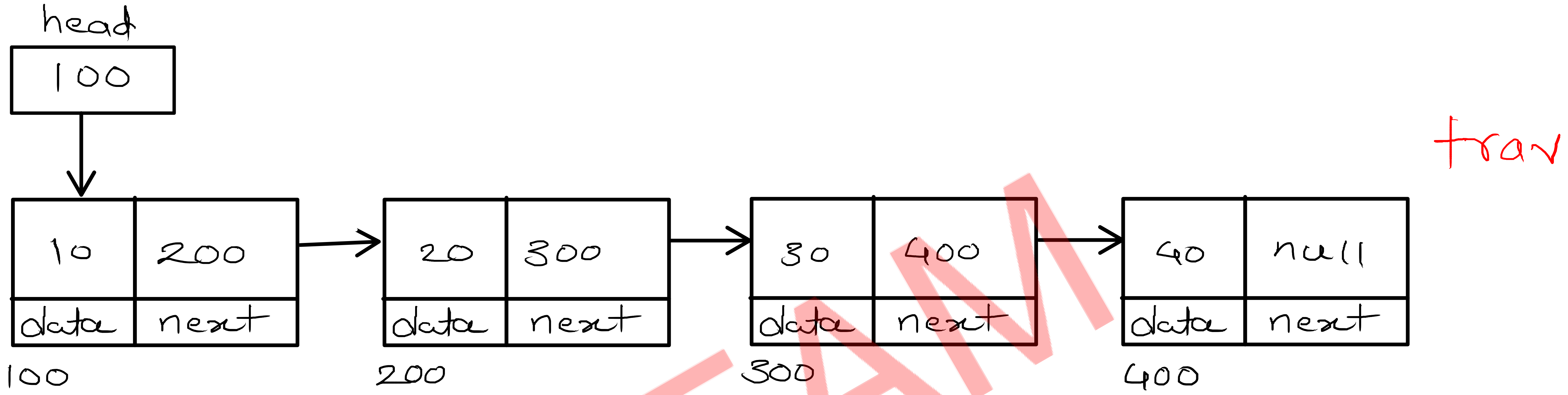




**//1. create a node with given data**
**//2. add head into next of newnode**
**//3. add newnode into head**

Time complexity : O(1)

# Singly Linear Linked List - Display



$T(n) = O(n)$

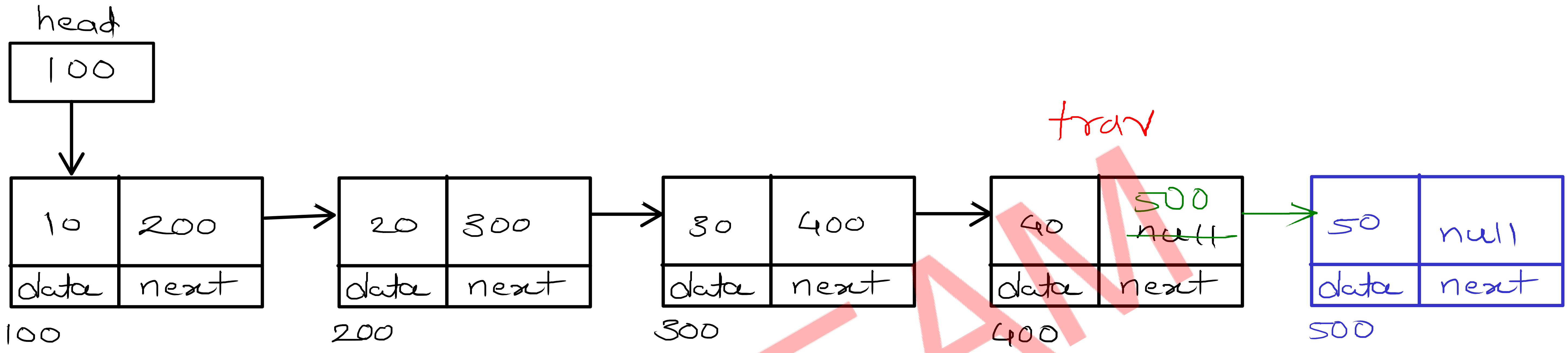
//1. create a trav referance and start at head

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

//3. go on next node (trav.next)

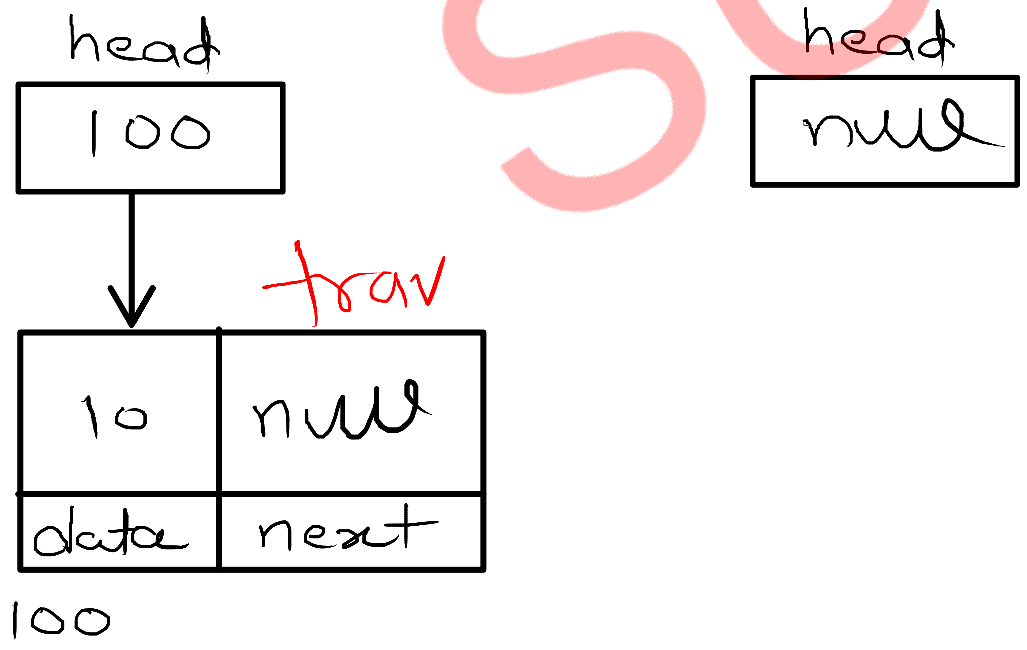
//4. repeat step 2 and 3 till last node

# Singly Linear Linked List - Add Last



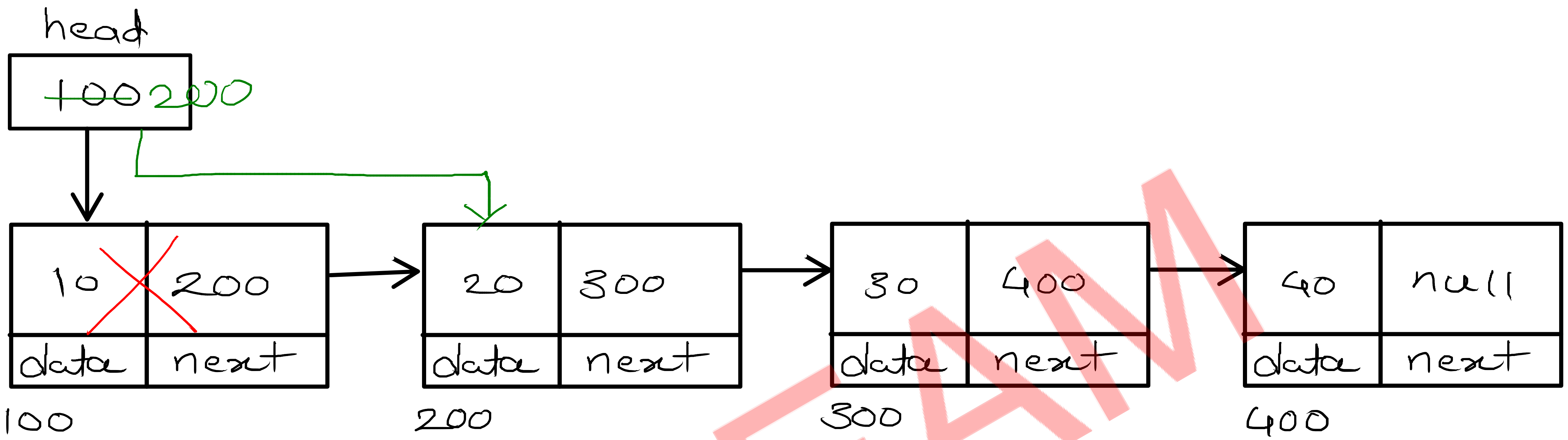
```
Node trav= head;
while(trav.next != null)
      trav=trav.next;
```

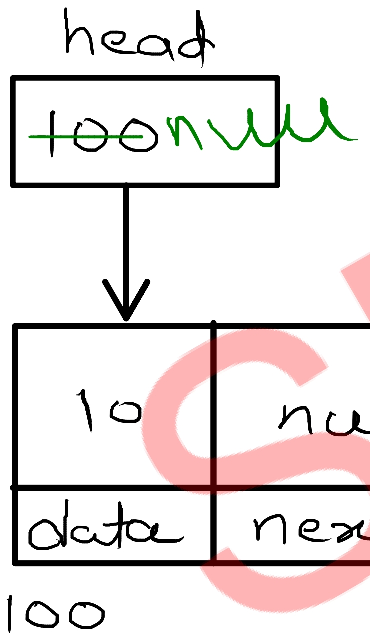
```
//1. create a newnode for given data
//2. if list is empty
        // add newnode into head itself
//3. if list is not wmpty
        //a. traverse till last node
        //b. add newnode into next of last node
```



$T(n) = O(n)$

# Singly Linear Linked List - Delete First

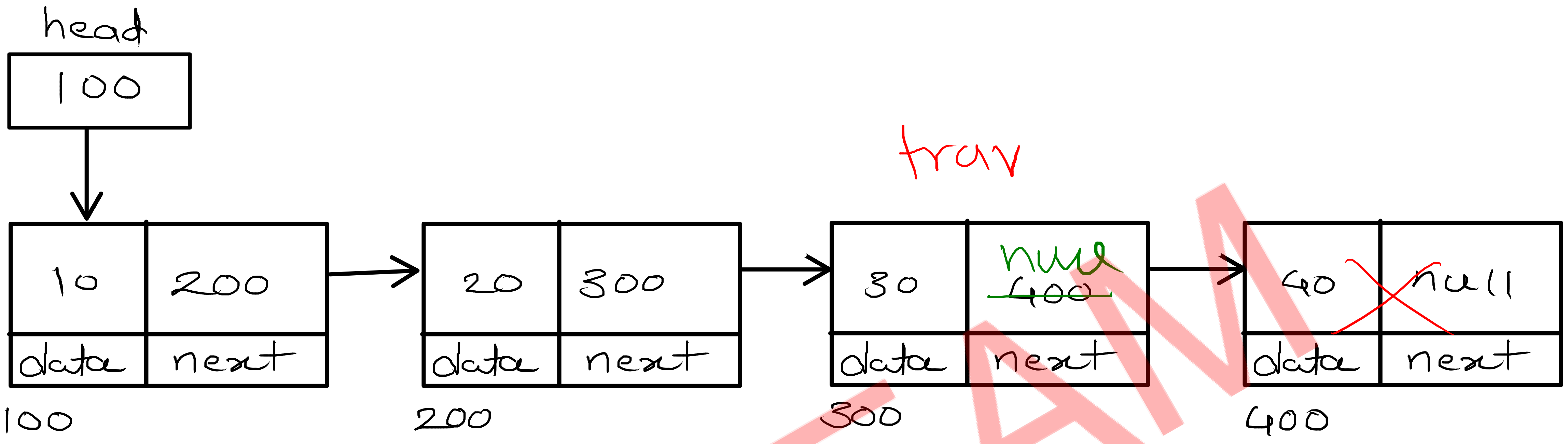




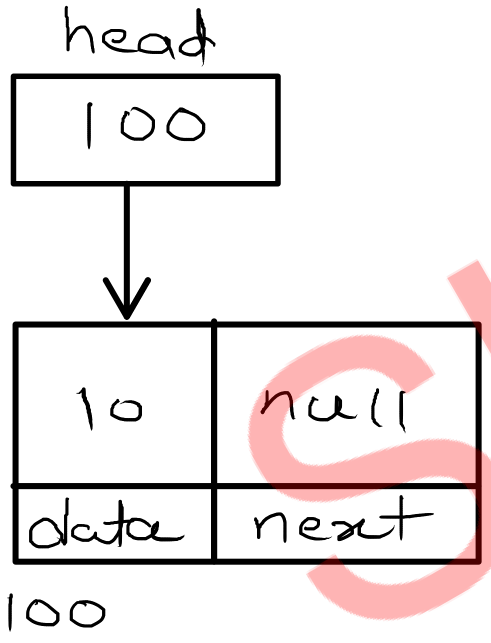
```
//1. if list is empty
        return;
//2. if list is not empty
        //a. move head on second node
```

$T(n) = O(1)$

# Singly Linear Linked List - Delete Last



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

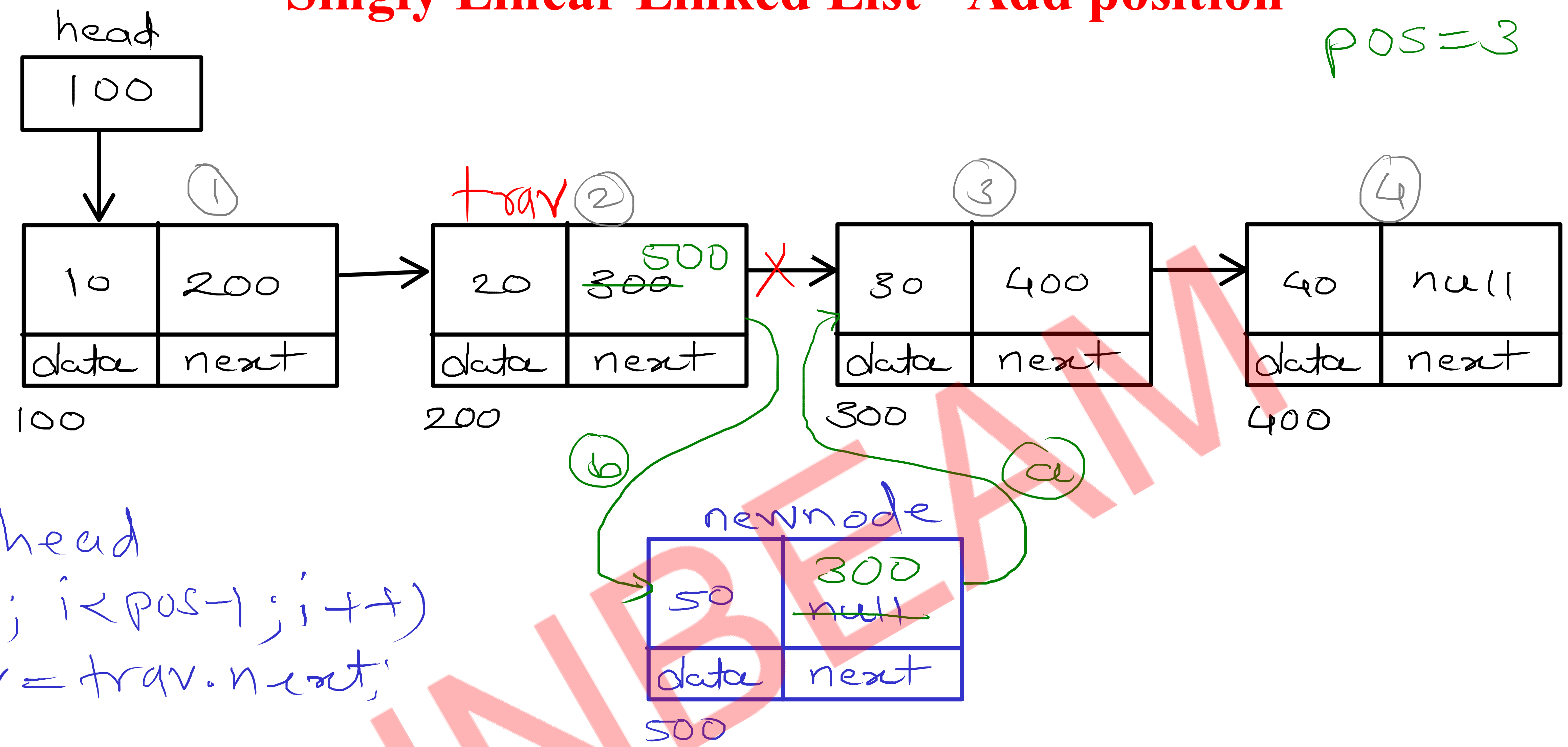


```
//1. if list is empty
        return;
//2. if list has single node
        head = null;
//3. if list has multiple node
        //a. traverse till second last node
        //b. add null into next of second last node
```

$T(n) = O(n)$

# Singly Linear Linked List - Add position

pos=3



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

pos=3

| trav | i | i<2 |
|---|---|---|
| 100 | 1 | T |
| 200 | 2 | F |

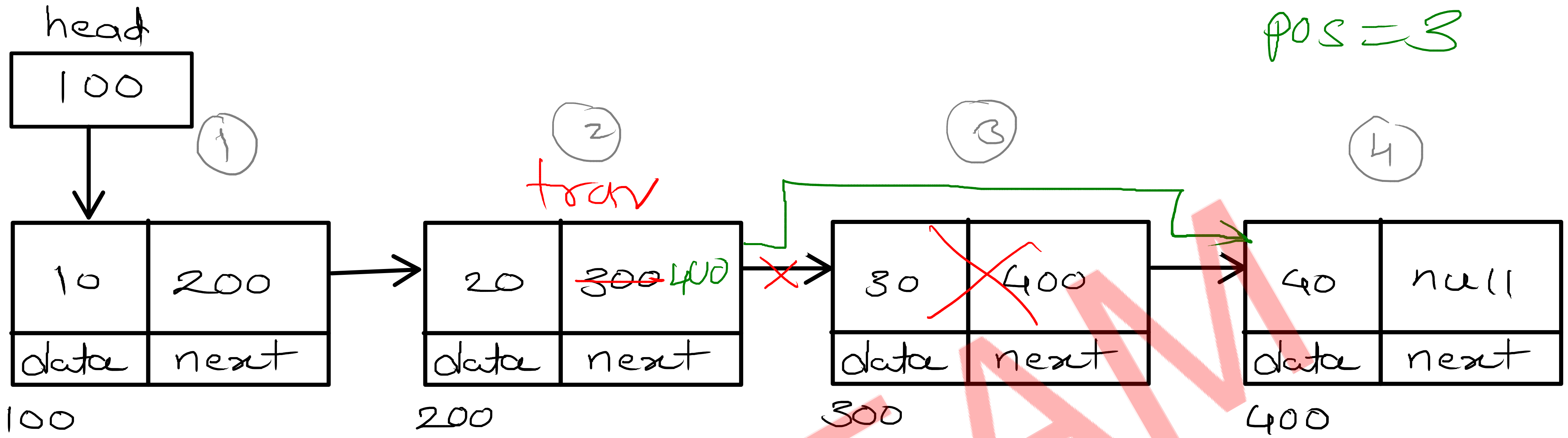
pos=5

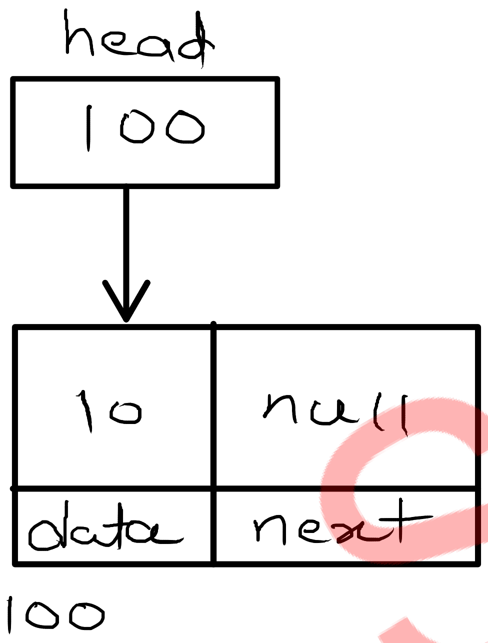
| trav | i | i<4 |
|---|---|---|
| 100 | 1 | T |
| 200 | 2 | T |
| 300 | 3 | T |
| 400 | 4 | F |

```
//1. create node with given data
//2. if list is empty
        // add newnode into head itself
//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
```

$T(n) = O(n)$

# Singly Linear Linked List - Delete position

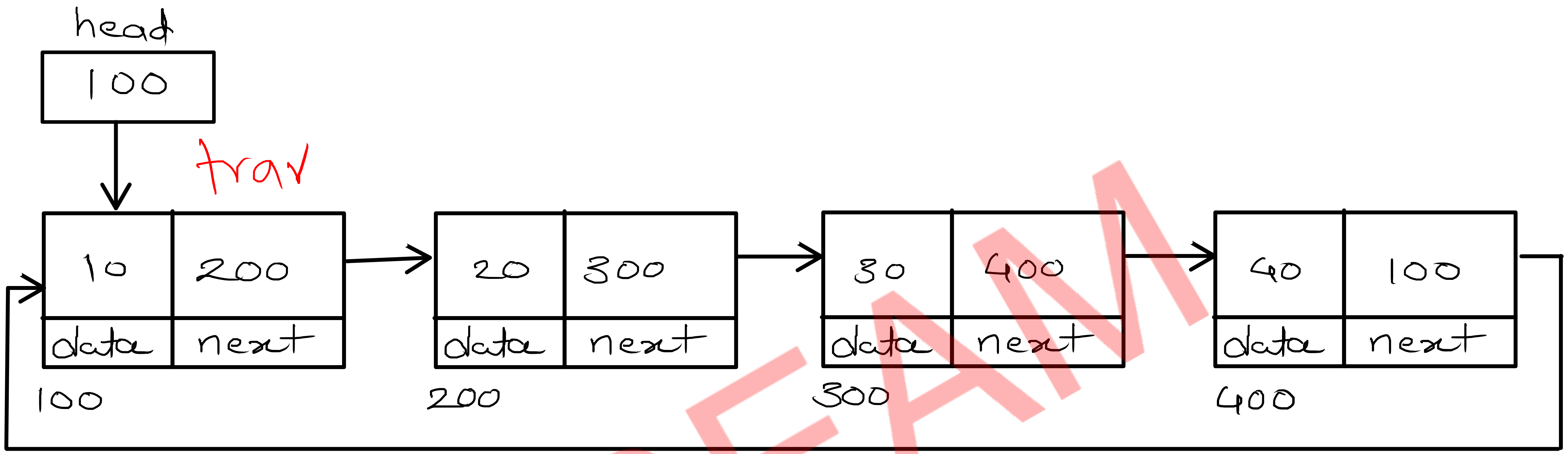




```
//1. if list is empty
        return;
//2. if list is not empty
        //a. traverse till pos - 1 node
        //b. add pos + 1 node into next of pos - 1
```

$T(n) = O(n)$

# Singly Circular Linked List - Display

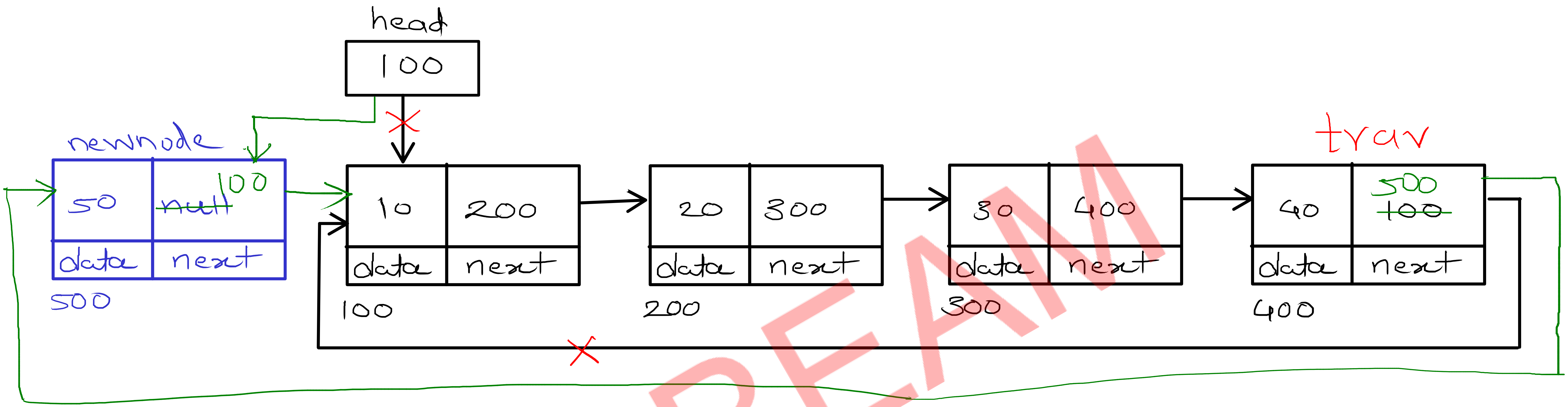


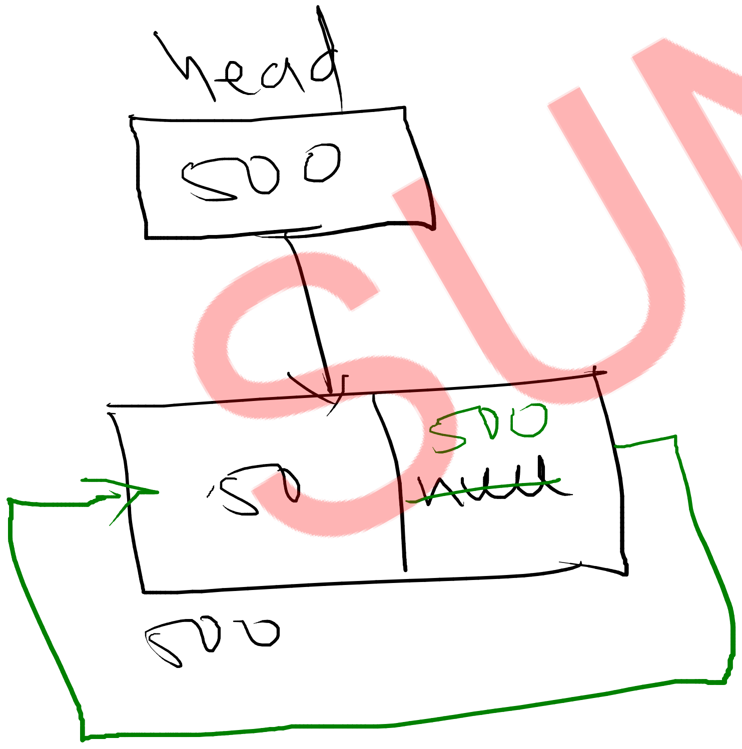
//1. create trav and start at head
//2. print/visit current node (trav.data)
//3. go on next node
//4. repeat step 2 and 3 till last node

$T(n) = O(n)$

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

# Singly Circular Linked List - Add First

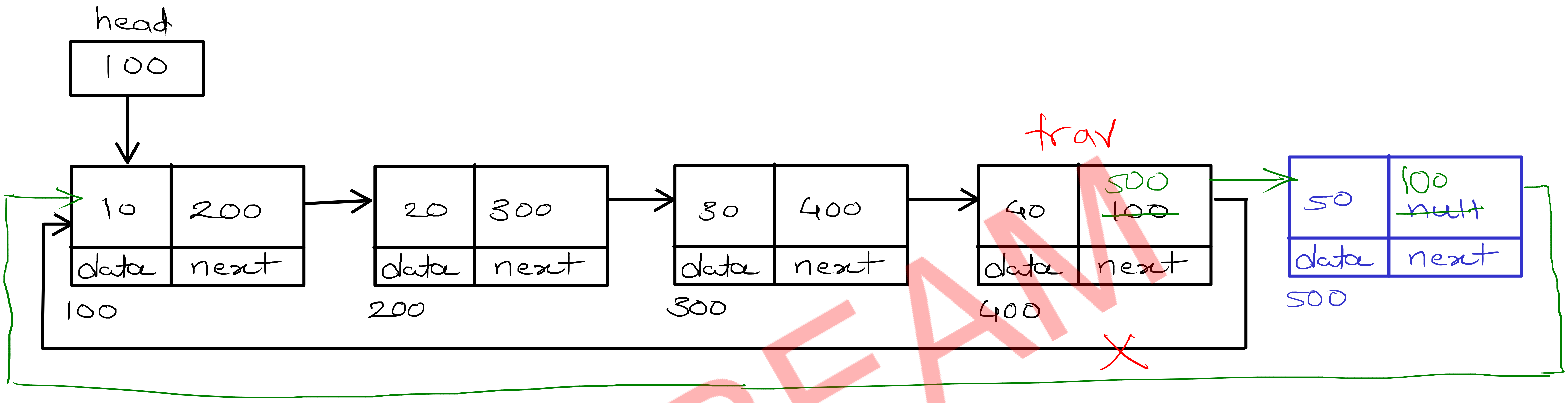




```
//1. create node with given data
//2. if list is empty
        //a. add newnode into head
        //b. make list circular
//3. if list is not empty
        //a. add first node into next of newnode
        //b. traverse till last  node
        //c. add newnode into next of last node
        //d. move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

T(n) = O(n)

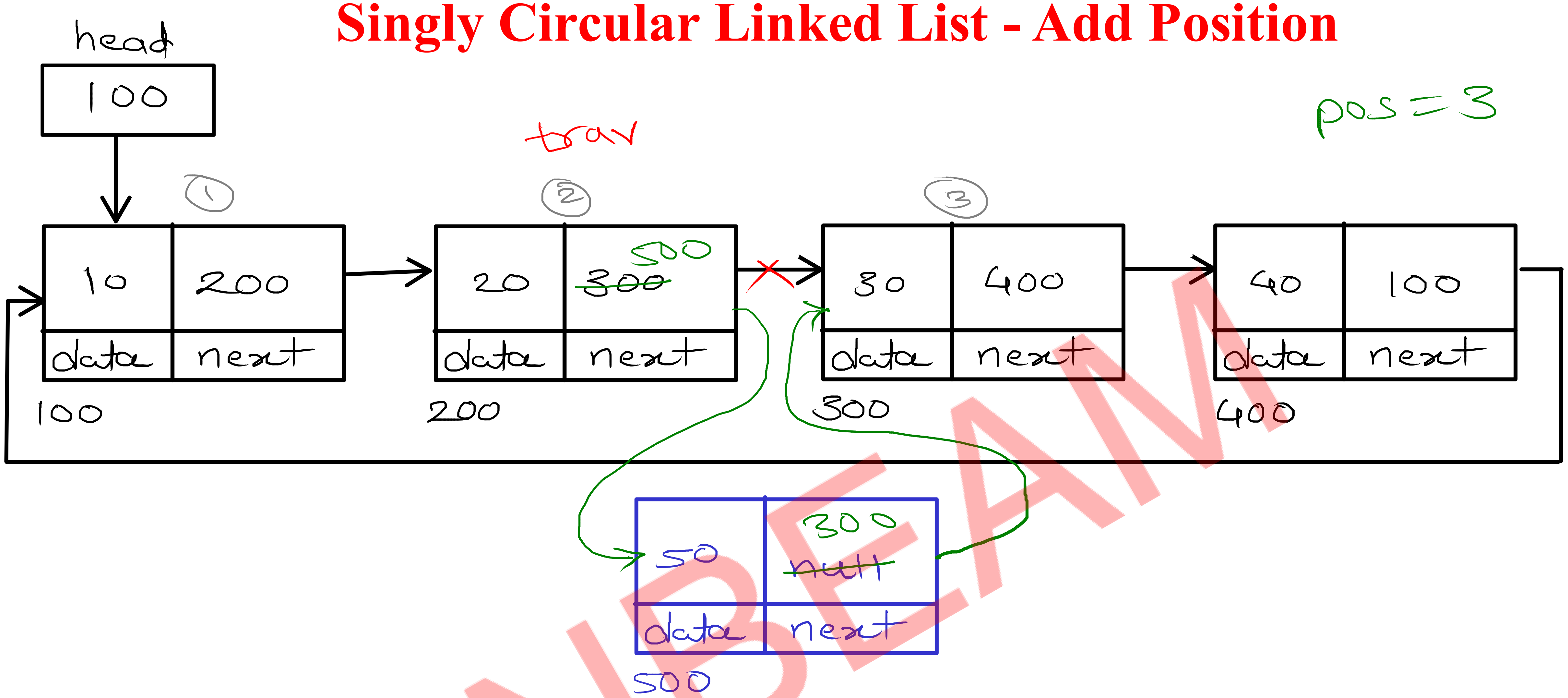
# Singly Circular Linked List - Add Last



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

$T(n) = O(n)$

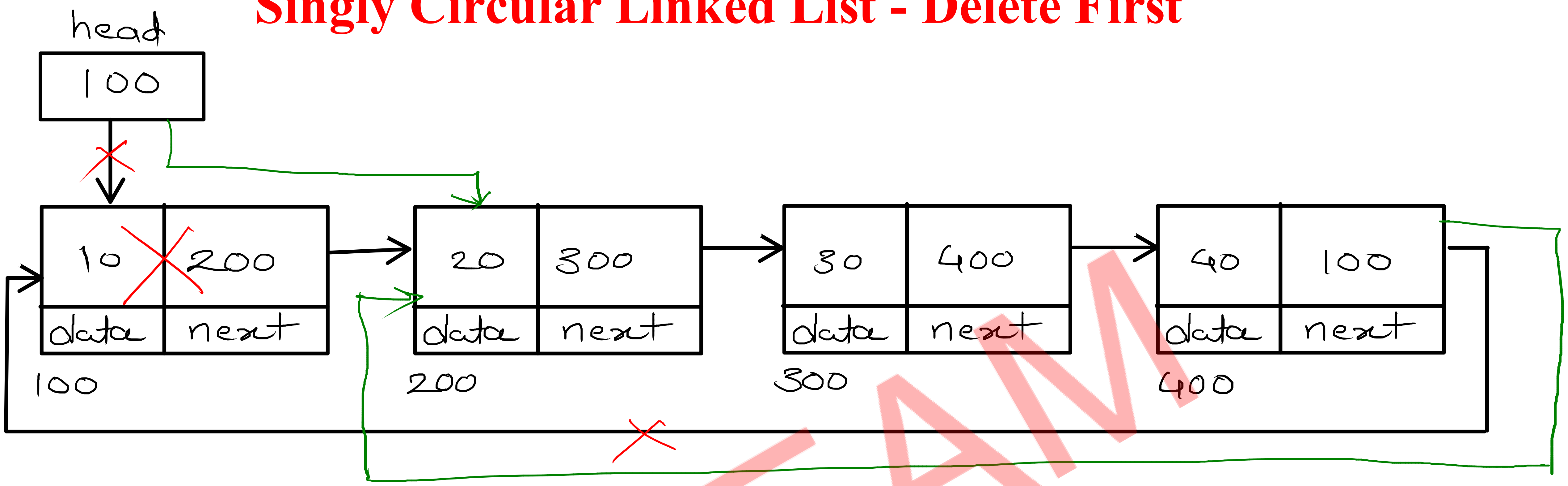
# Singly Circular Linked List - Add Position

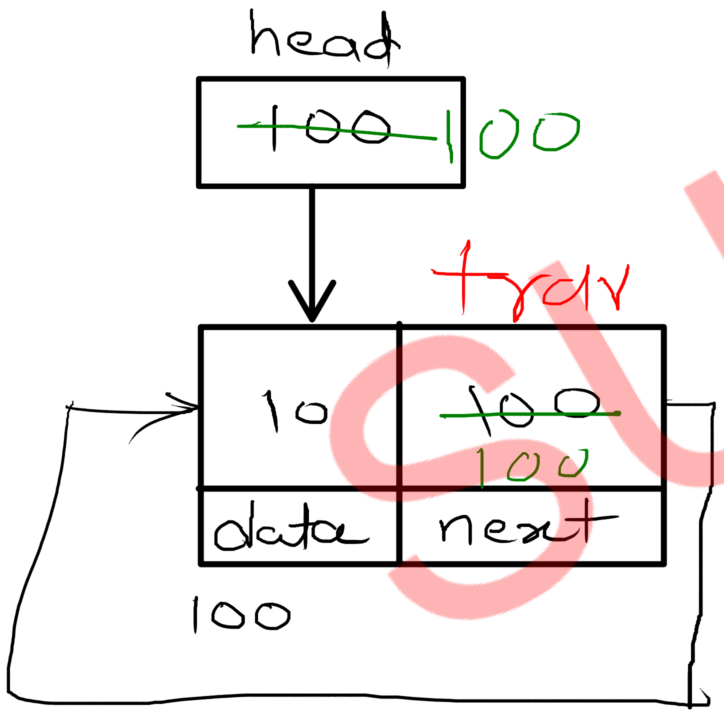


```
//1. create node
//2. if list is empty
        //a. add newnode into head
        //b. make it circular
//3. if list is not empty
        //a. traverse till pos-1
        //b. add pos node into next of newnode
        //c. add newnode into next of pos-1 node
```

$T(n) = O(n)$

# Singly Circular Linked List - Delete First

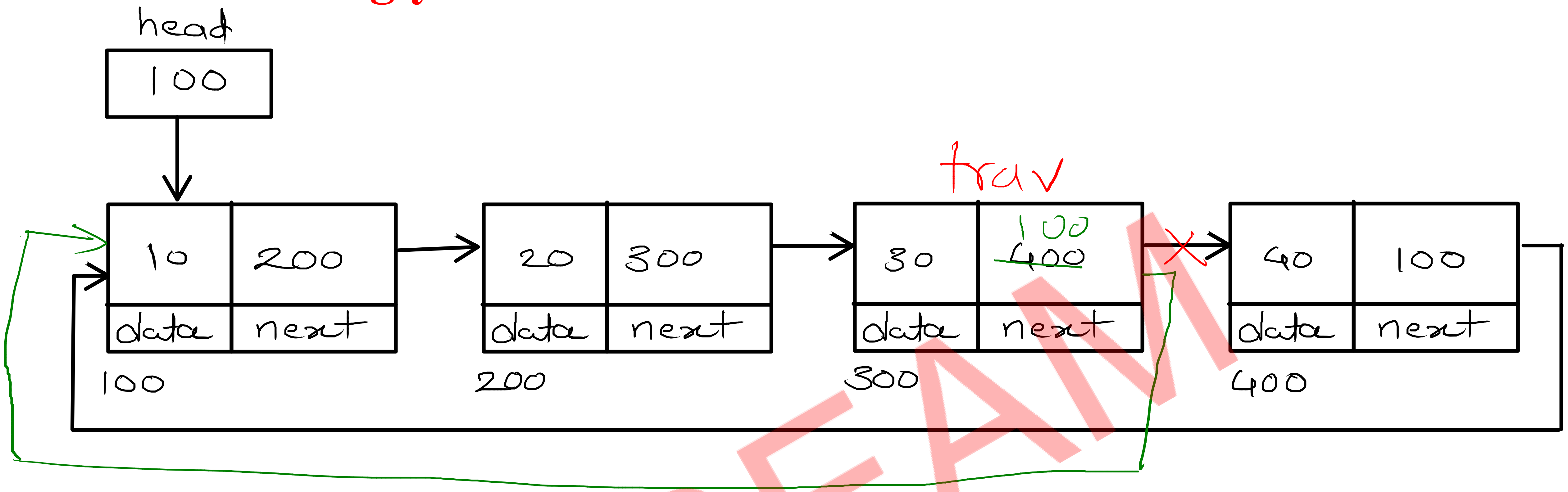




```
//1. if list is empty
        return;
//2. if list has single node
        // make head = null
//3. if list has multiple nodes
        //a. traverse till last node
        //b. add second node into next of last node
        //c. move head on second node
```

$T(n) = O(n)$

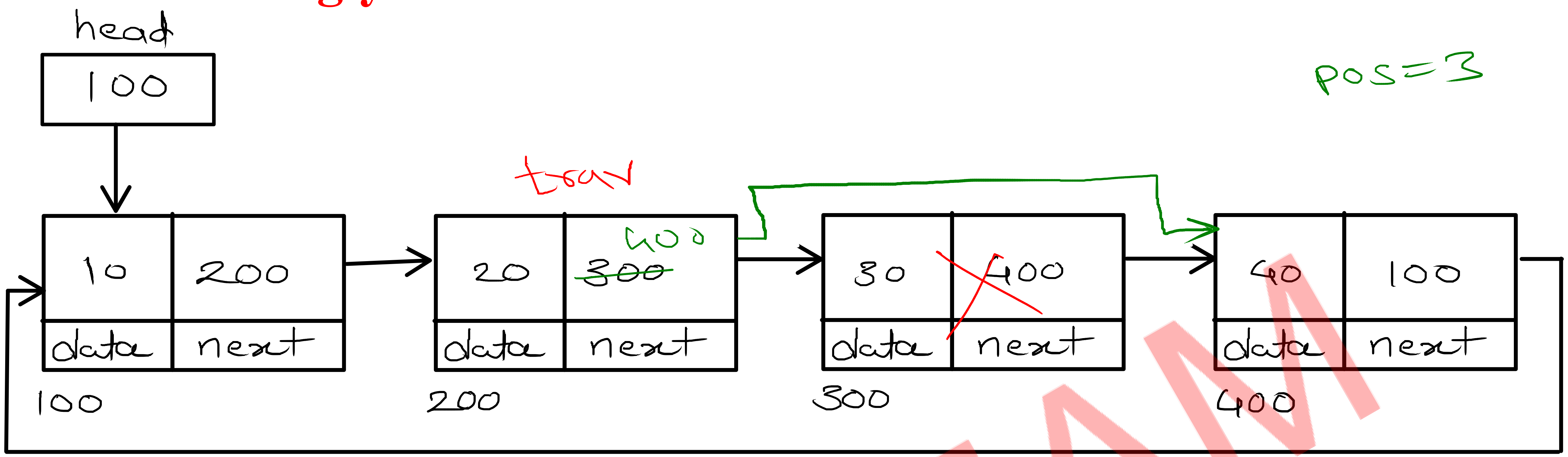
# Singly Circular Linked List - Delete Last



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

$T(n) = O(n)$

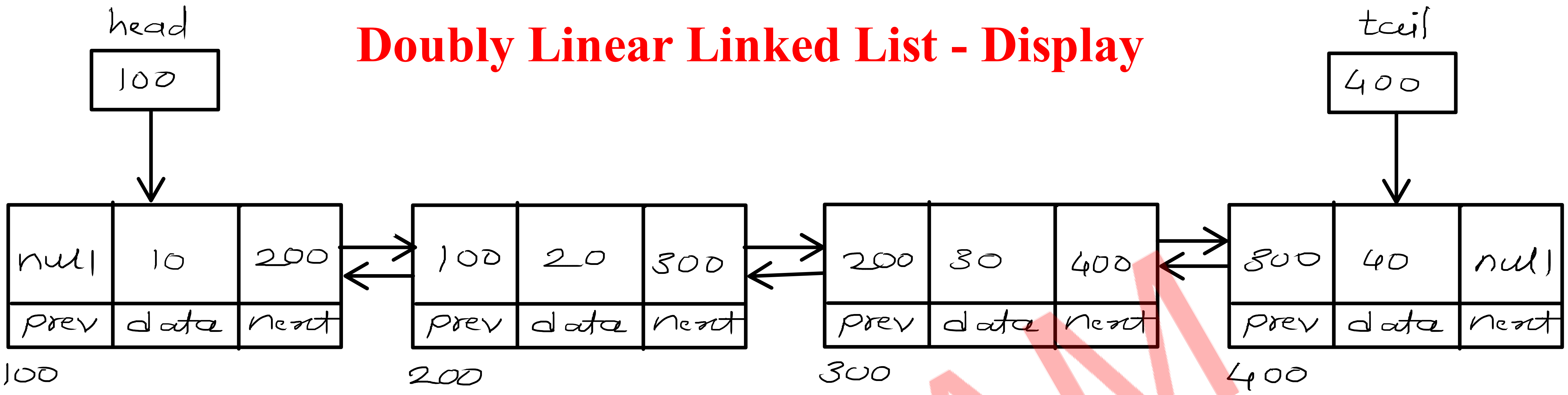
# Singly Circular Linked List - Delete Position



```
//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)

# Doubly Linear Linked List - Display

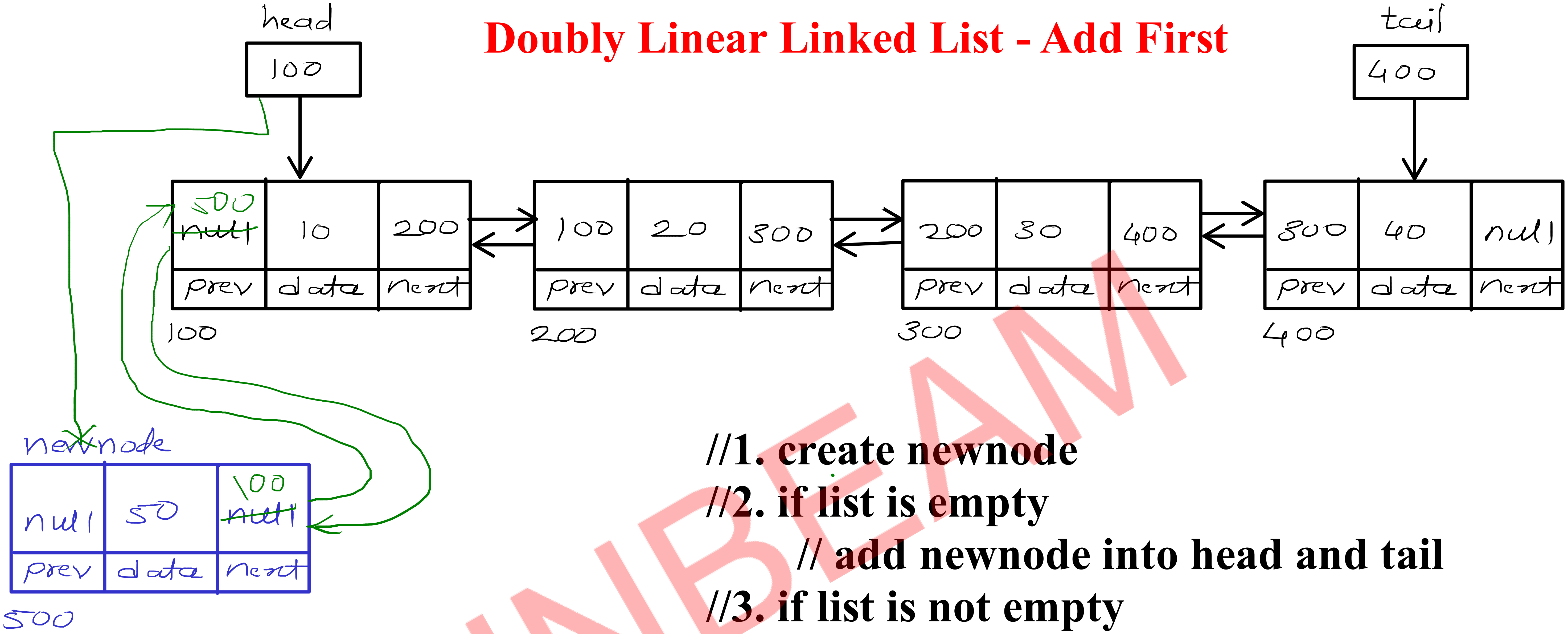


// forward list
//1. create a trav pointer and start at head
//2. print current node
//3. go on next node
//4. repeat step 2 and 3 till last node

// backward list
//1. create a trav pointer and start at tail
//2. print current node
//3. go on prev node
//4. repeat step 2 and 3 till first node

$T(n) = O(n)$

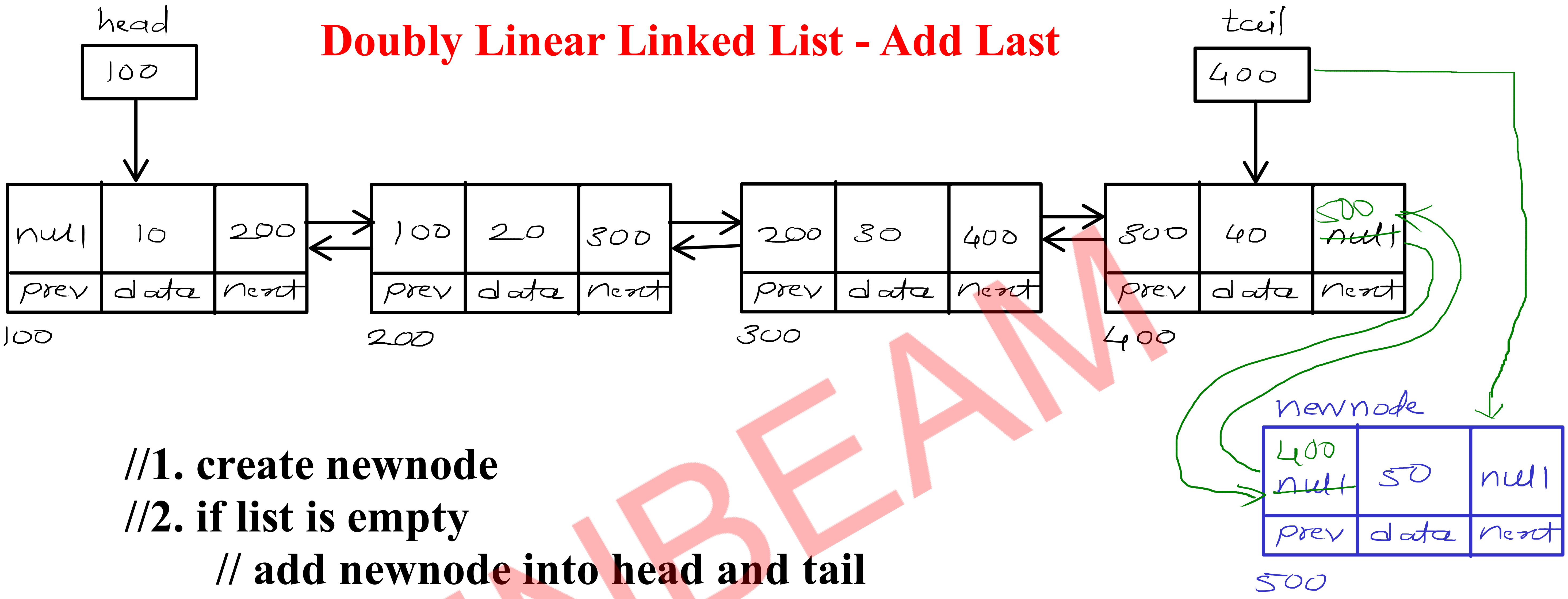
# Doubly Linear Linked List - Add First



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

$T(n) = O(1)$

# Doubly Linear Linked List - Add Last



//1. create newnode

//2. if list is empty

        // add newnode into head and tail

//3. if list is not empty

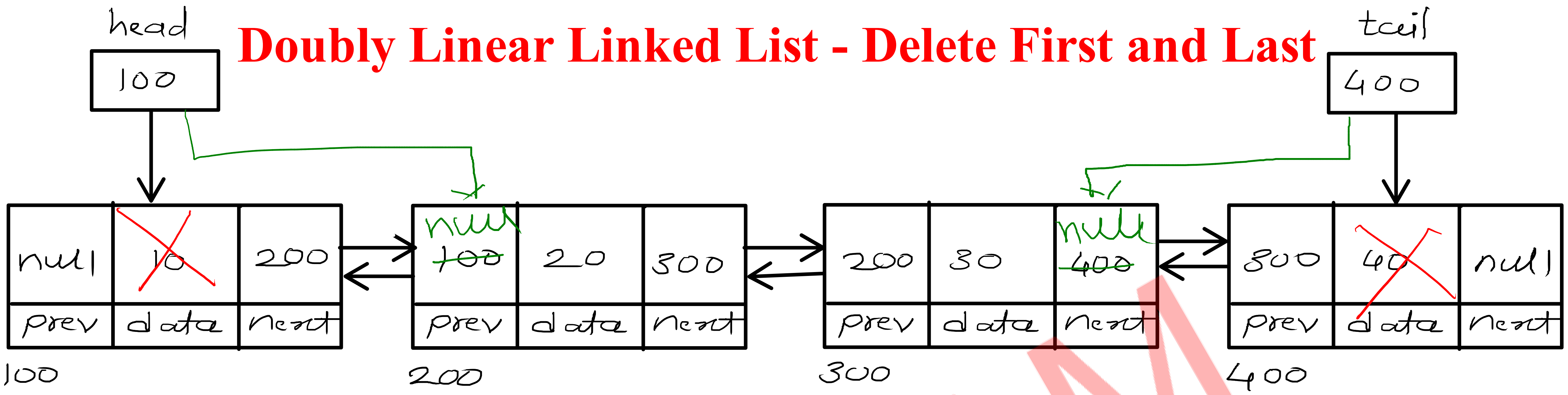
        //a. add last node into prev of newnode

        //b. add newnode into next of last node

        //c. move tail on newnode

$T(n) = O(1)$

# Doubly Linear Linked List - Delete First and Last



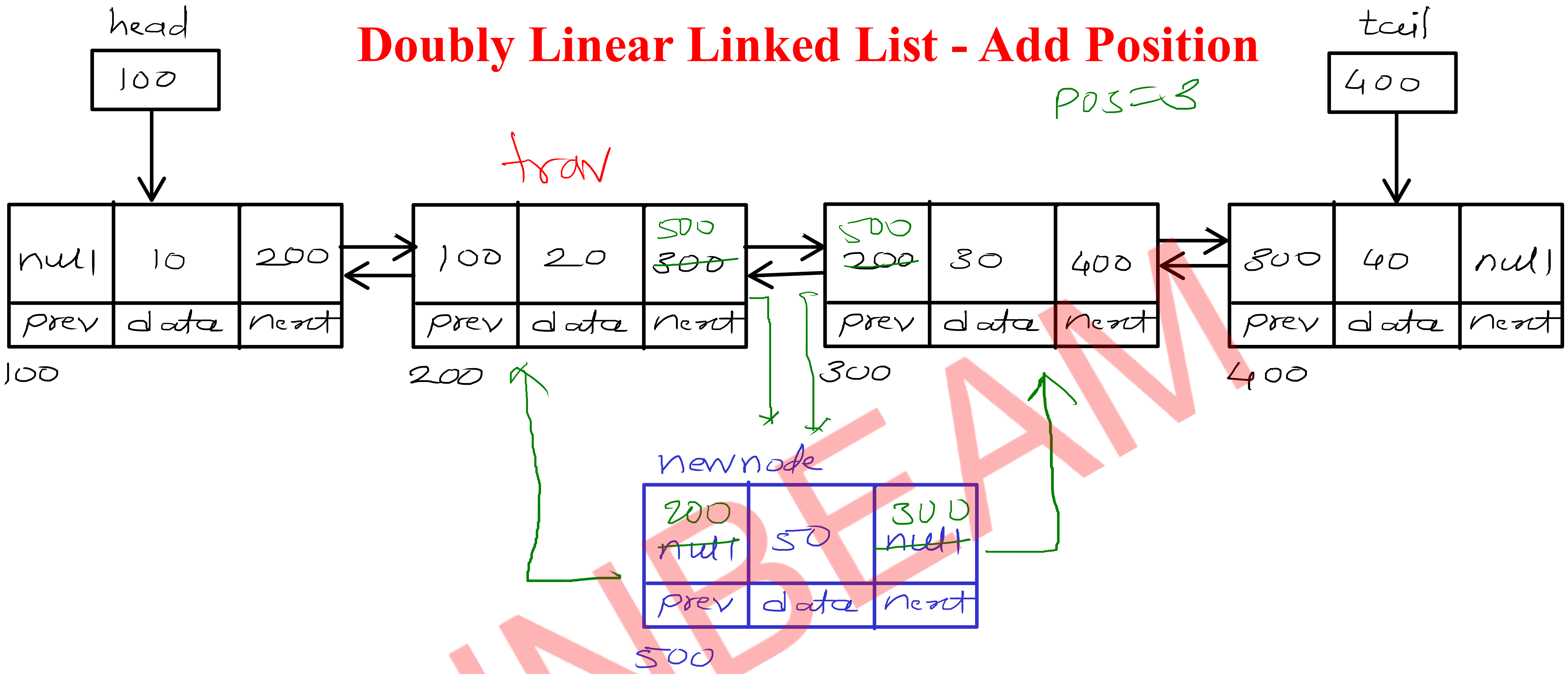
```
//1. if list is empty
        return;
//2. if list has single node
        head = tail = null;
//3. if list has multiple nodes
        //a. add null into prev of second node
        //c. move head on second node
```

$T(n) = O(1)$

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

$T(n) = O(1)$

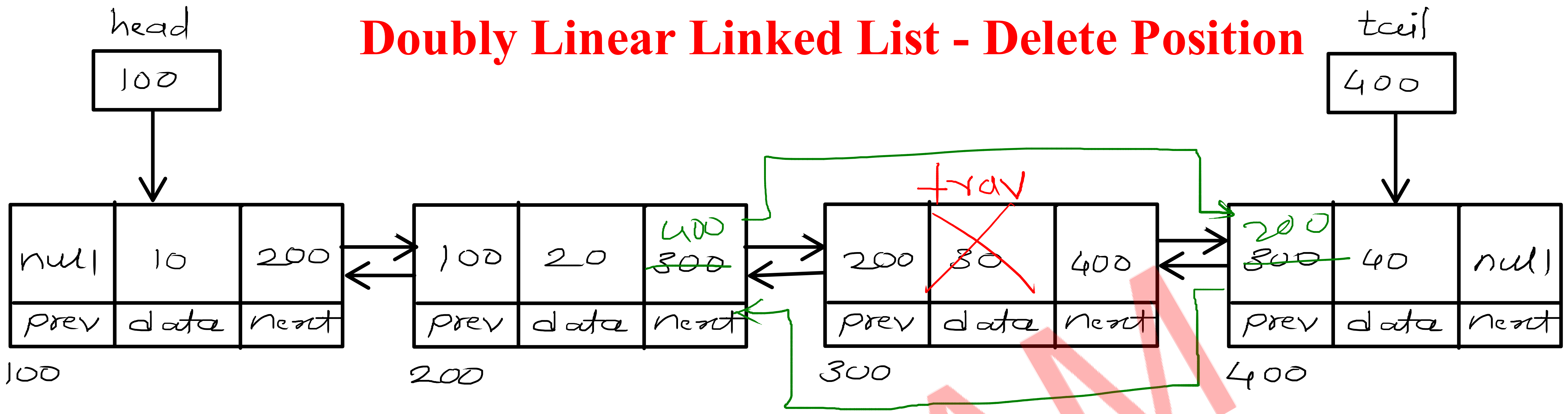
# Doubly Linear Linked List - Add Position



```
//1. create newnode
//2. if list is empty
        // add newnode into head and tail
//3. if list is not empty
        //a. traverse till pos -1 node
        //b. add pos node into next of newnode
        //c. add pos -1 node into prev of newnode
        //d. add newnode into prev of pos node
        //e. add newnode into next of pos -1 node
```

$T(n) = O(n)$

# Doubly Linear Linked List - Delete Position



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

$T(n) = O(n)$