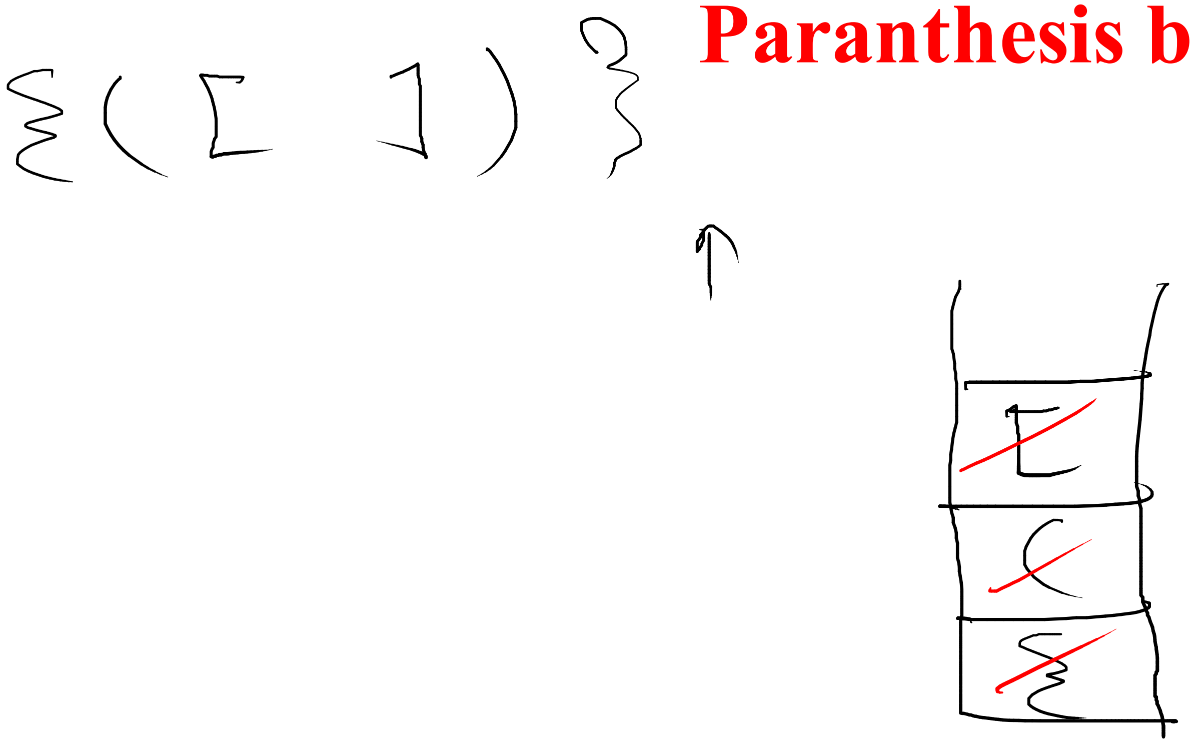
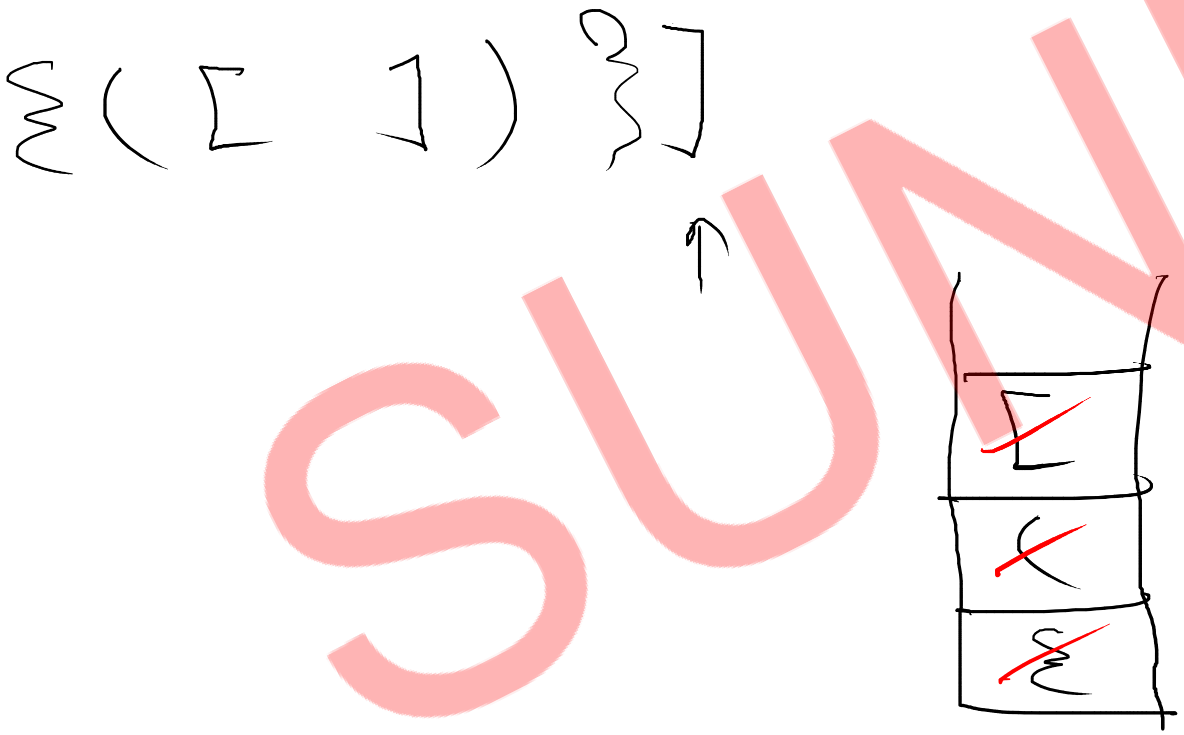
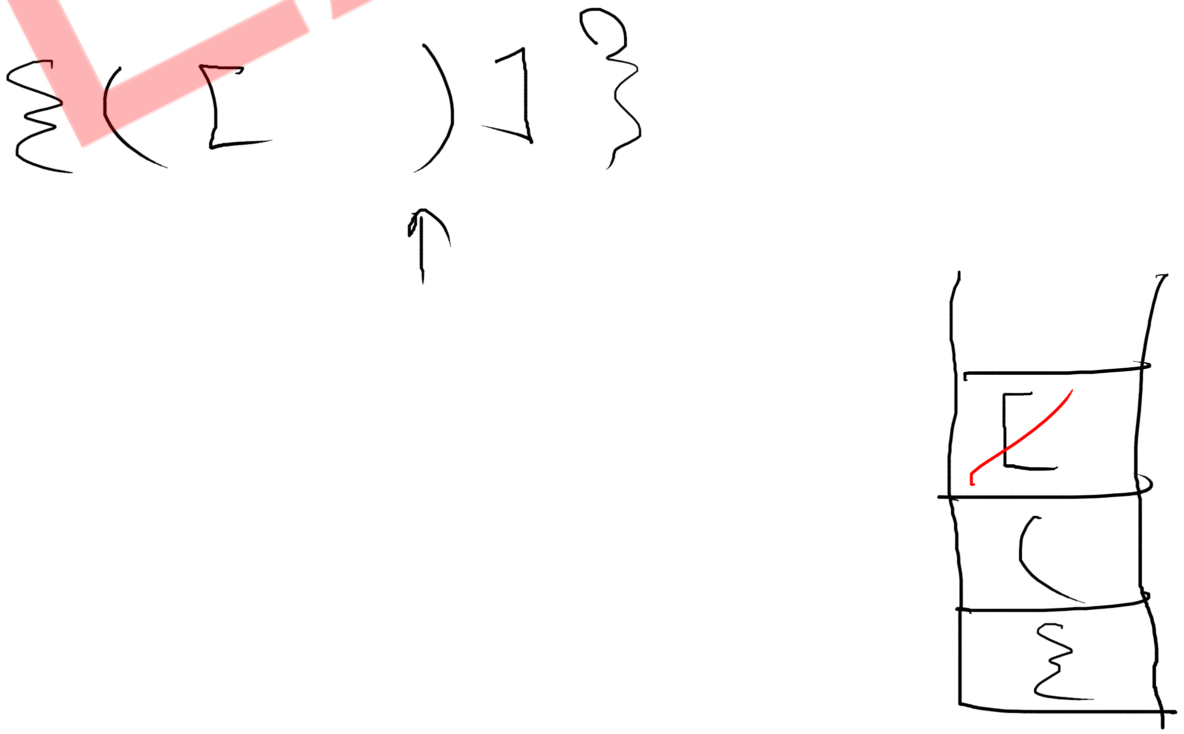
Paranthesis balancing using stack

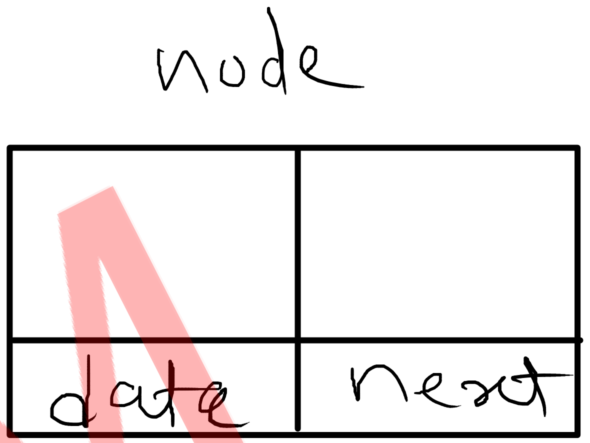


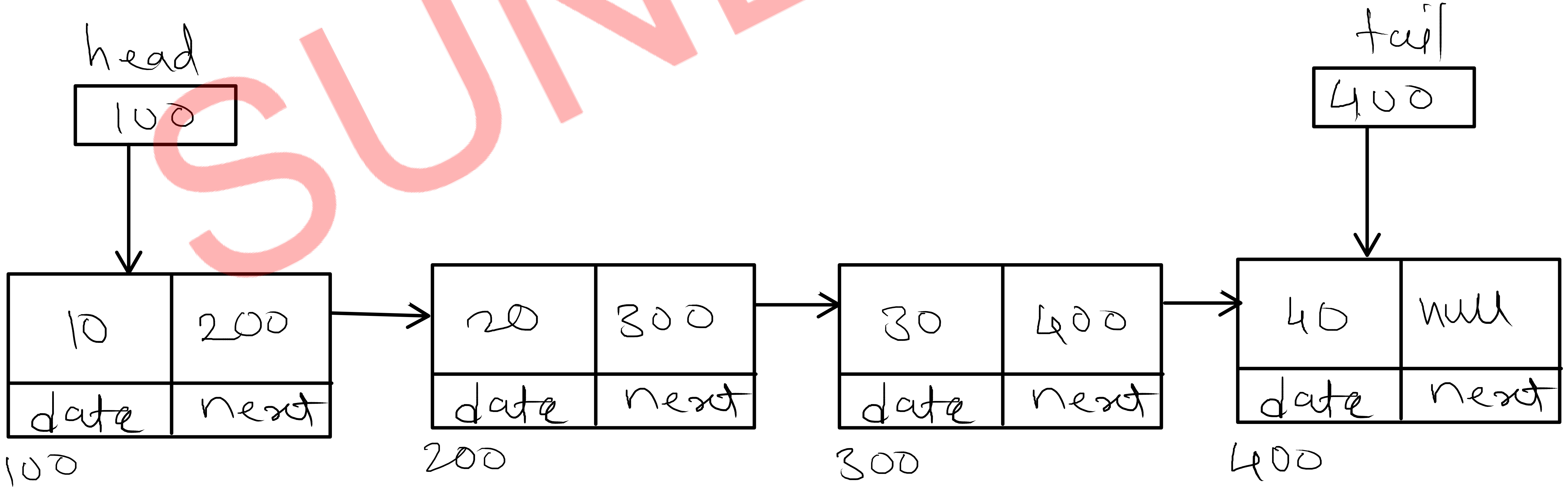




Linked List

- linear data structure (data is arranged sequentially)
- link of next data is kept with previous data
- every data element of linked list is known as "node"
- node consist of two parts:
  1. data - actual data
  2. link - address of next data (node)
- address of first node is kept into referance known as "head"
- address of last node is kept into referance known as "tail" (optional)





Operations

1. Add first
2. Add last
3. Add position (insert)

4. Delete first
5. Delete last
6. Delete position (remove)

7. Display (traverse)

8. Search
9. Sort
10. Reverse
11. Mid

Types

1. Singly Linear Linked List
2. Singly Circular Linked List
3. Doubly Linear Linked List
4. Doubly Circular Linked List

```
        class Node{
                type data;
                Node next;
        }

class List{
        static class Node{
                type data;
                Node next;
        }
        Node head;
        Node tail;
        int count;
        public List()   {}
        public Add_first() {}
        public Add_Last()           {}
        public Delete_first()       {}
        public Delete_last()        {}
        public display()            {}
}
```

→ int, char, float, double, enum, user defined class

self referential class :

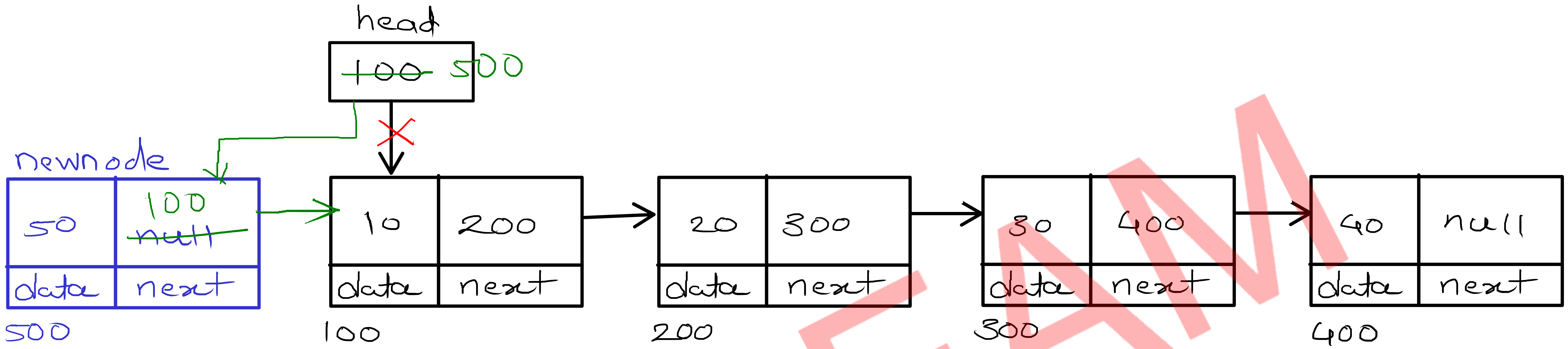
↳ reference of same type is kept into class.

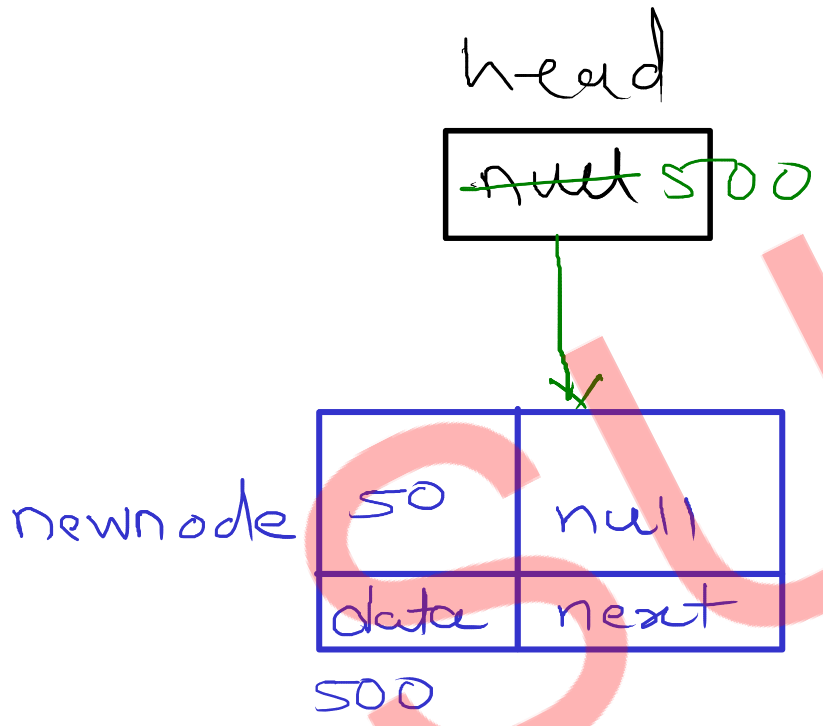
```
class List {
  → static class Node {}
    Node head;
    ...
    class Iterator {
        trav;
        Iterator() {
            trav = head;
        }
    }
}
```

① no dependency of List class to create object of Node class

② non static fields of outer class are not directly accessible into inner class

Singly Linear Linked List - Add First





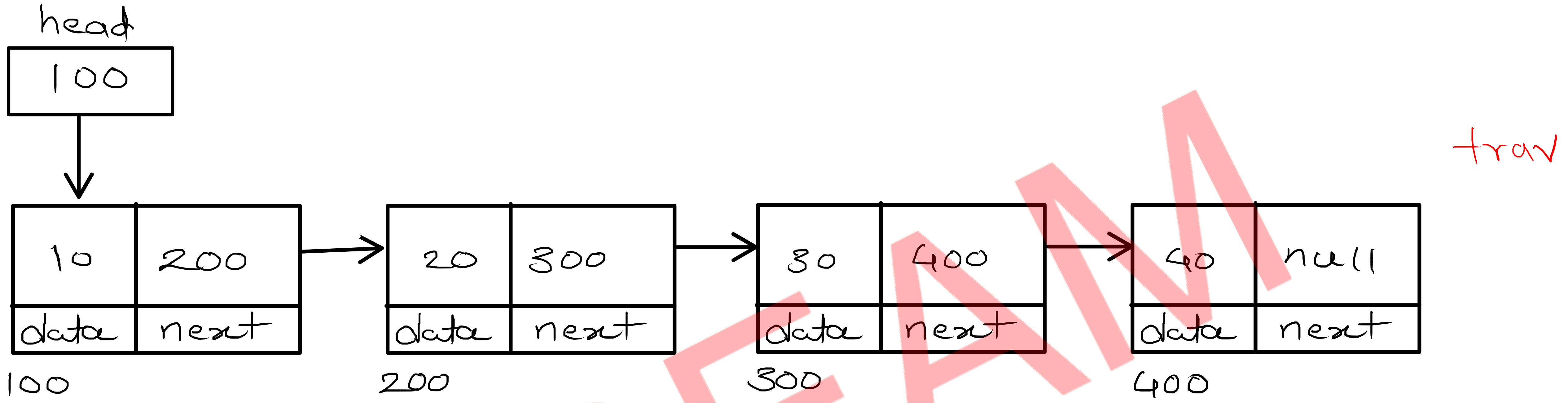
//1. create newnode for given value

//2. add first node into next of newnode

//3. move head on newnode

$T(n) = O(1)$

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

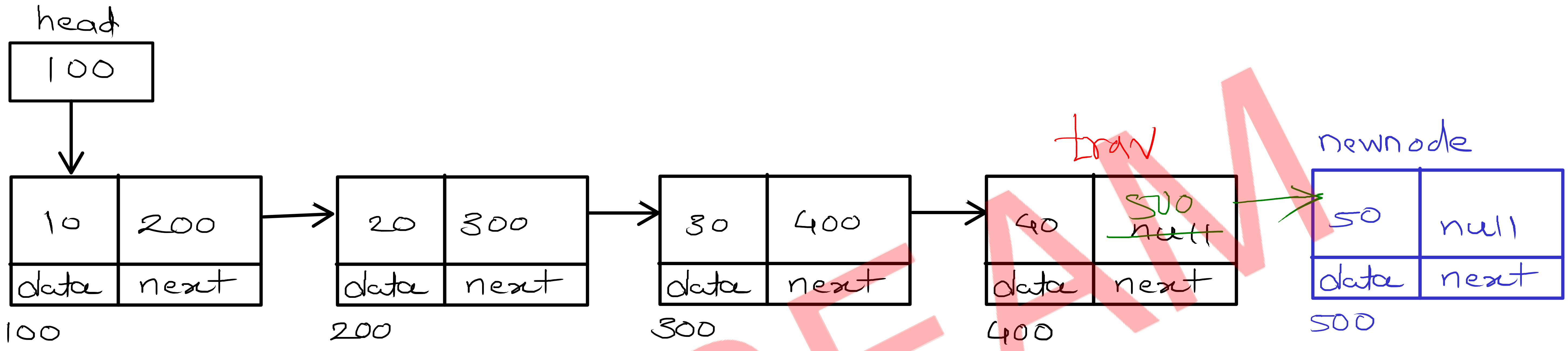
//2. visit/print data of current node (trav.data)

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

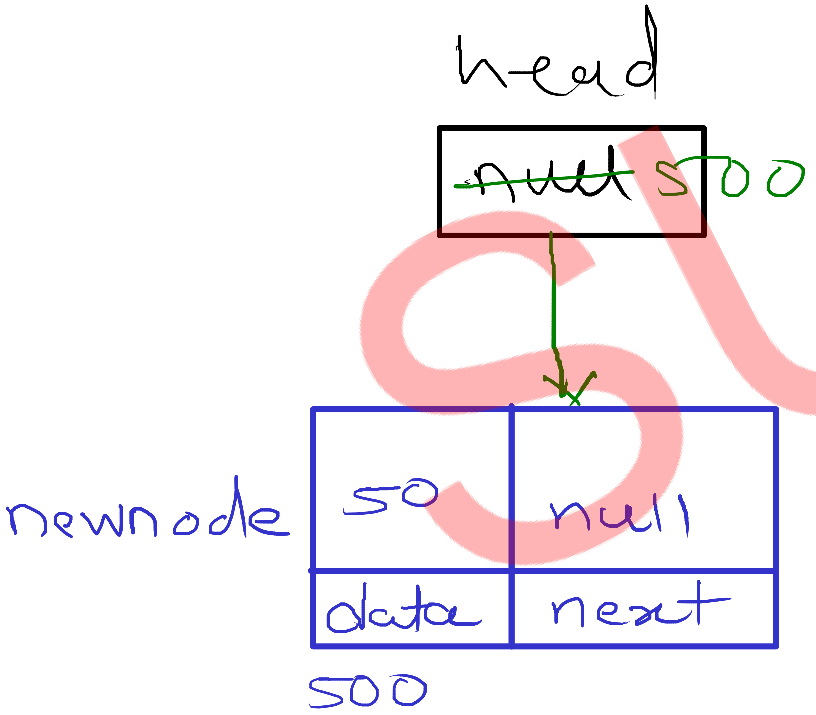
//4. repeat step 2 and 3 till last node of list

$T(n) = O(n)$

Singly Linear Linked List - Add Last



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

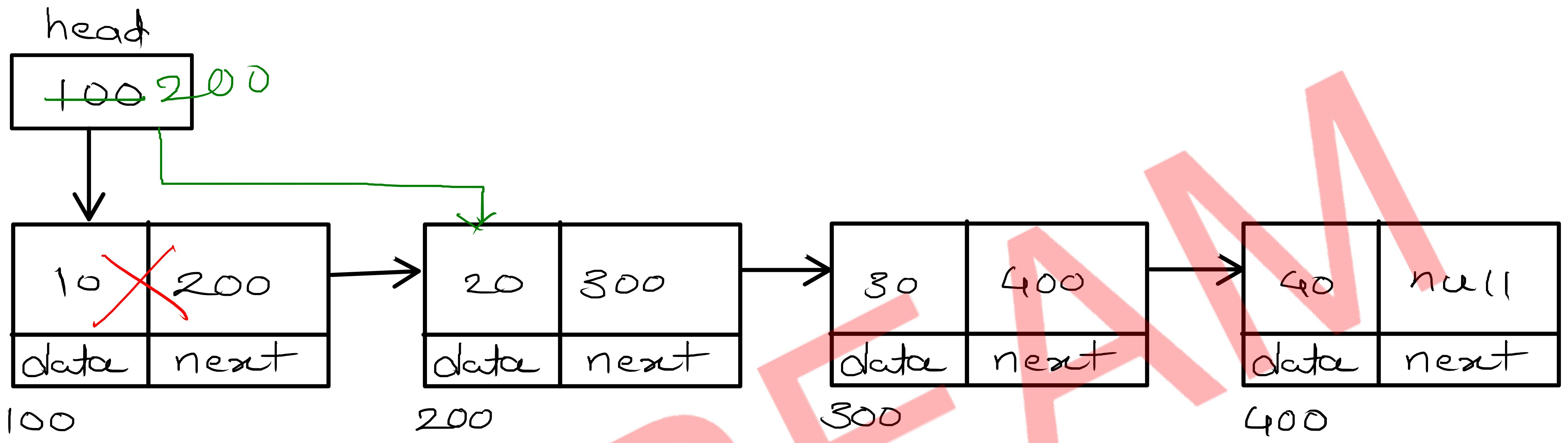


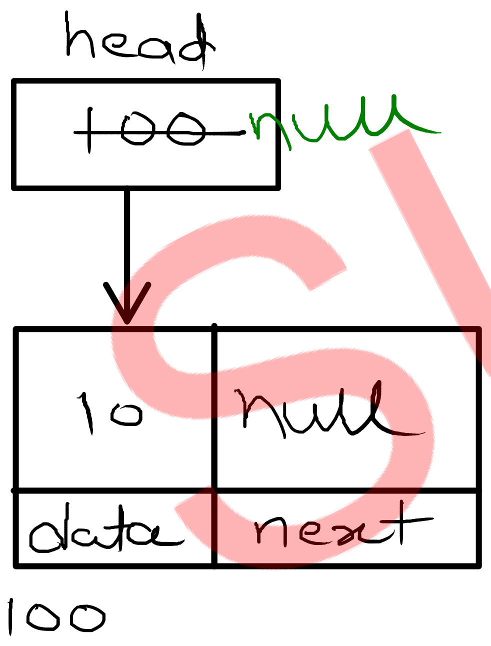
```
while(trav.next != null)
    trav = trav.next;
```

$T(n) = O(n)$

trav
100
200
300
400

Singly Linear Linked List - Delete First

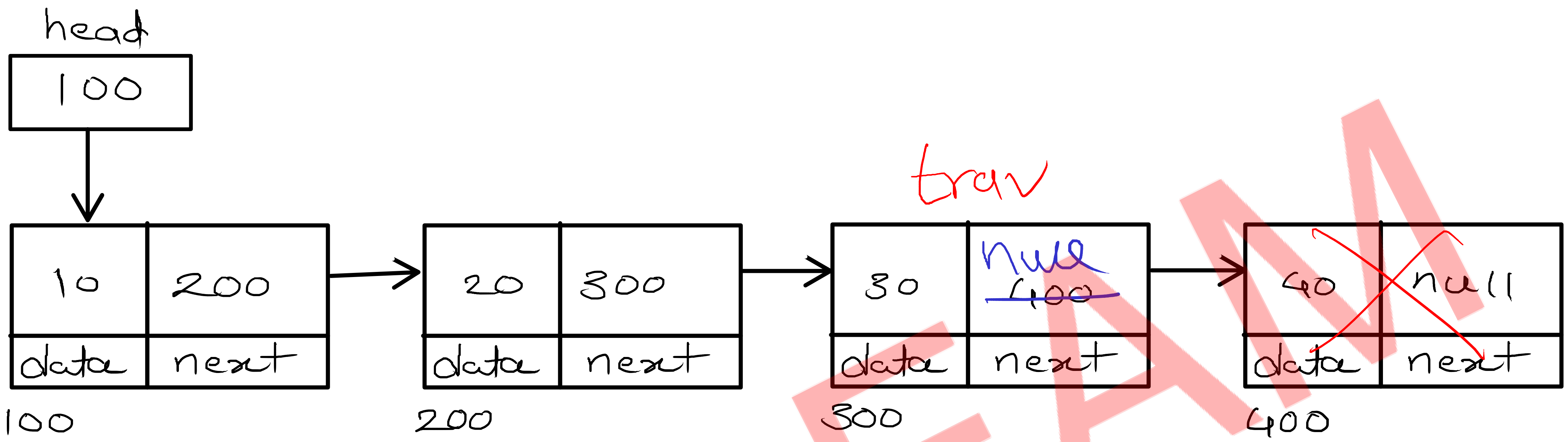


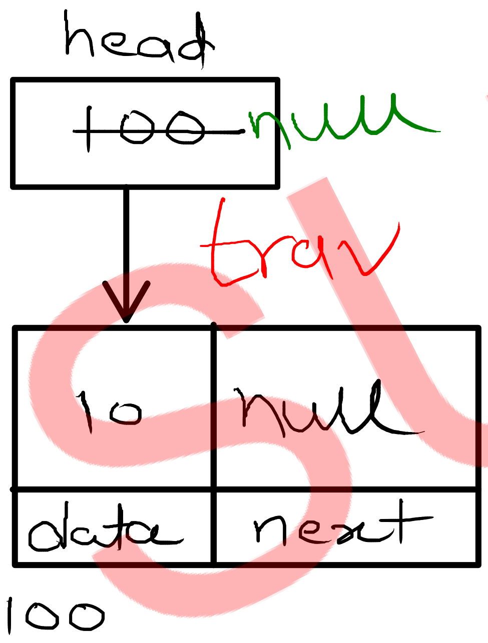


```
//0. if list is empty
        return;
//1. if list is not empty
        //a. move head on second node
```

$T(n) = O(1)$

Singly Linear Linked List - Delete Last





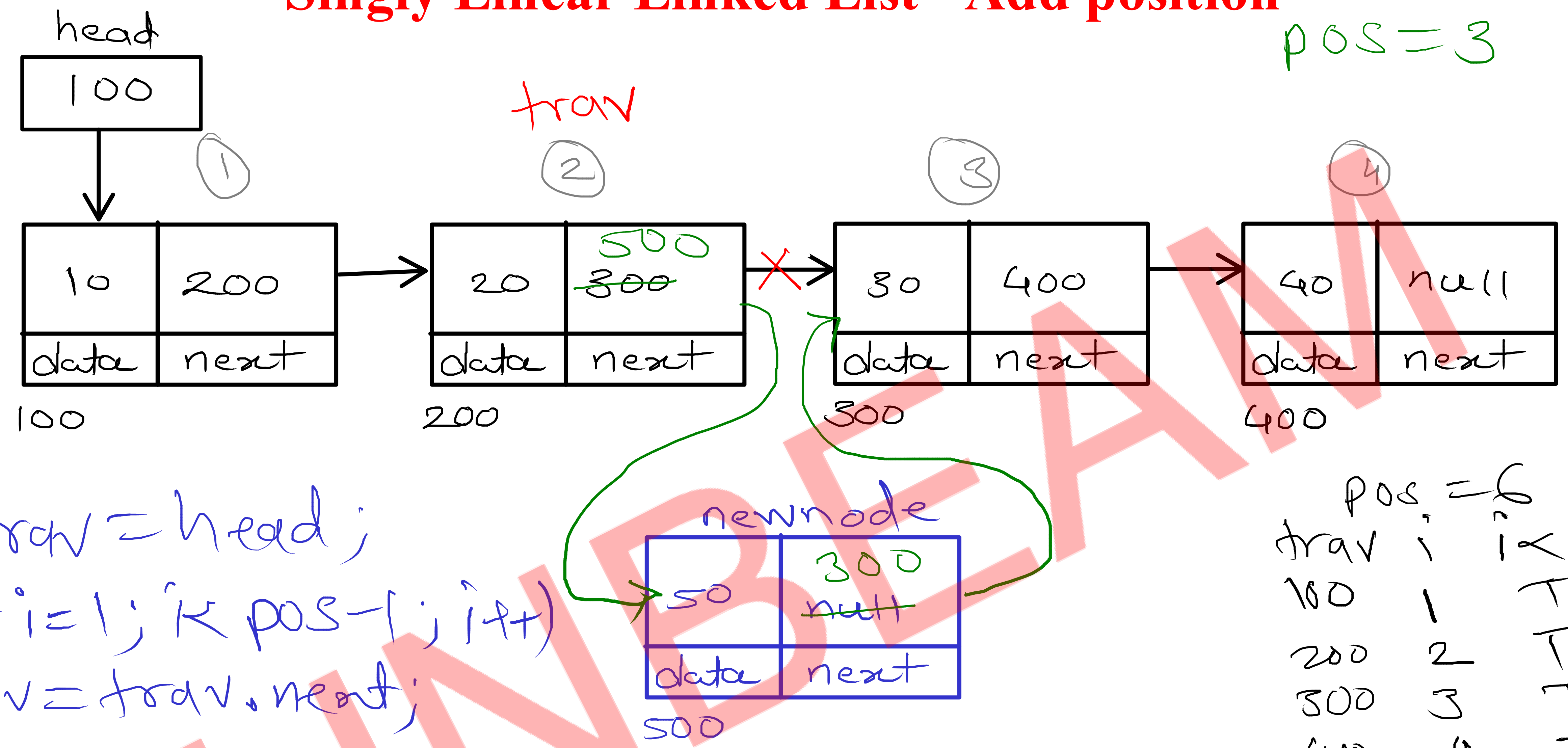
```
//1. if list is empty
        return;
//2. if list has single node
        head = null;
//3. if list has multiple node
        //a. traverse till second last node
        //b. add null into next of second last node
```

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

$T(n) = O(n)$

Make before break

Singly Linear Linked List - Add position



```
Node trav = head;
for(int 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	

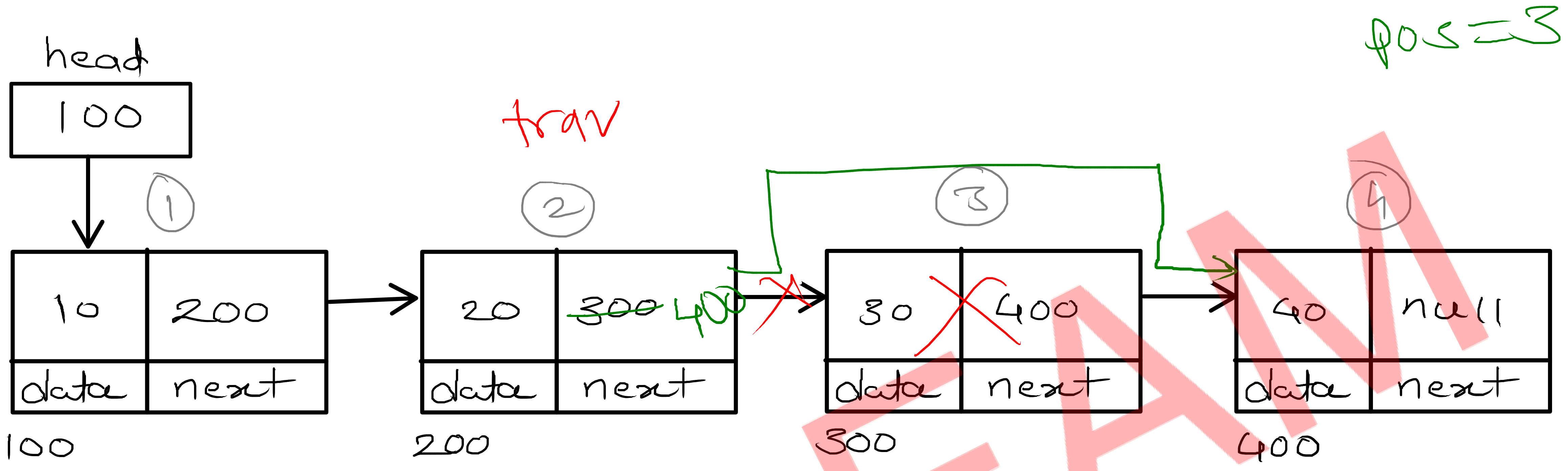
pos=6

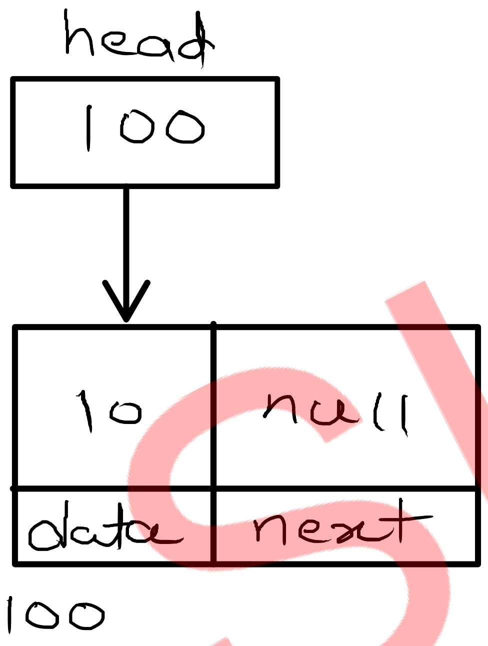
trav	i	i<5
100	1	T
200	2	T
300	3	T
400	4	T
null	5	F

$T(n) = O(n)$

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

Singly Linear Linked List - Delete position

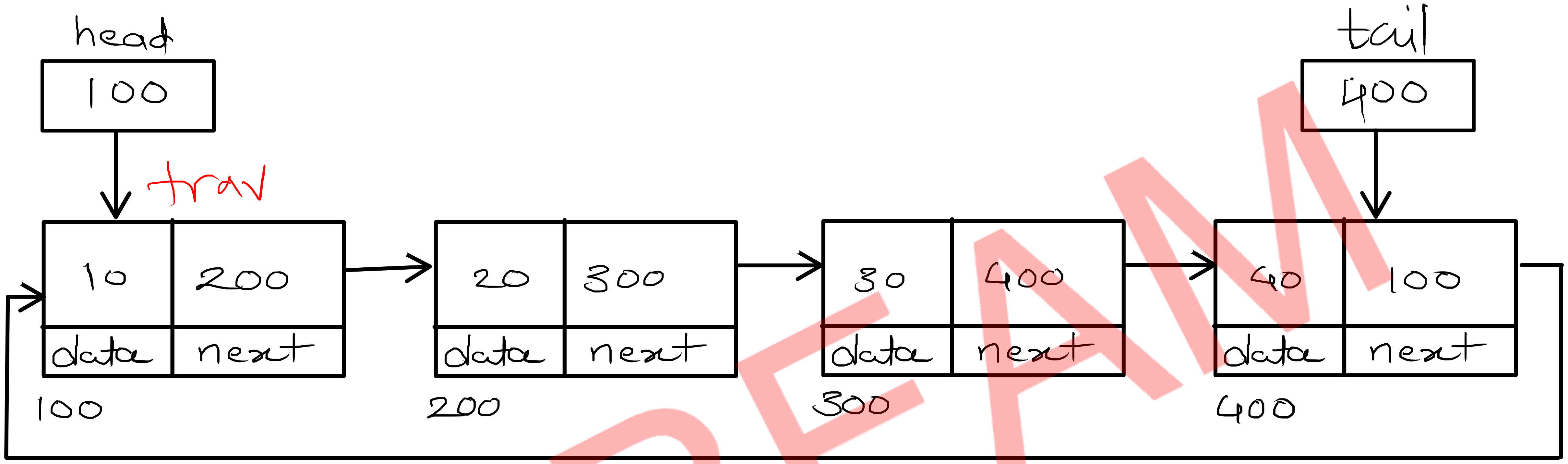




```
//1. if list is empty
        return;
//2. if list has single node
        head = null;
//3. if list has multiple nodes
        //a. traverse till pos -1 node
        //b. add pos+1 node into next of pos-1 node
```

$T(n) = O(n)$

Singly Circular Linked List - Display

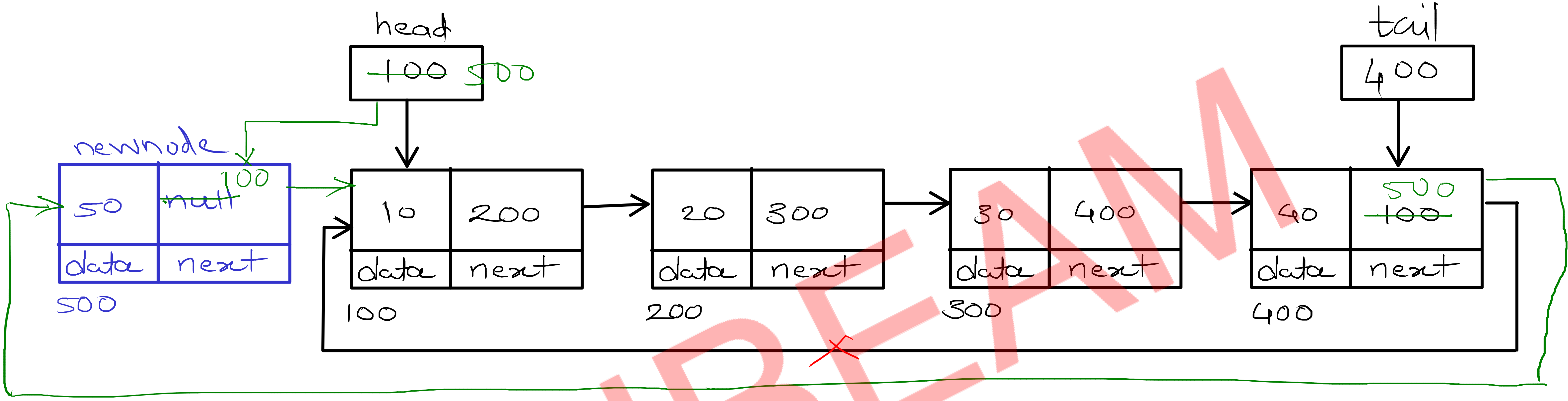


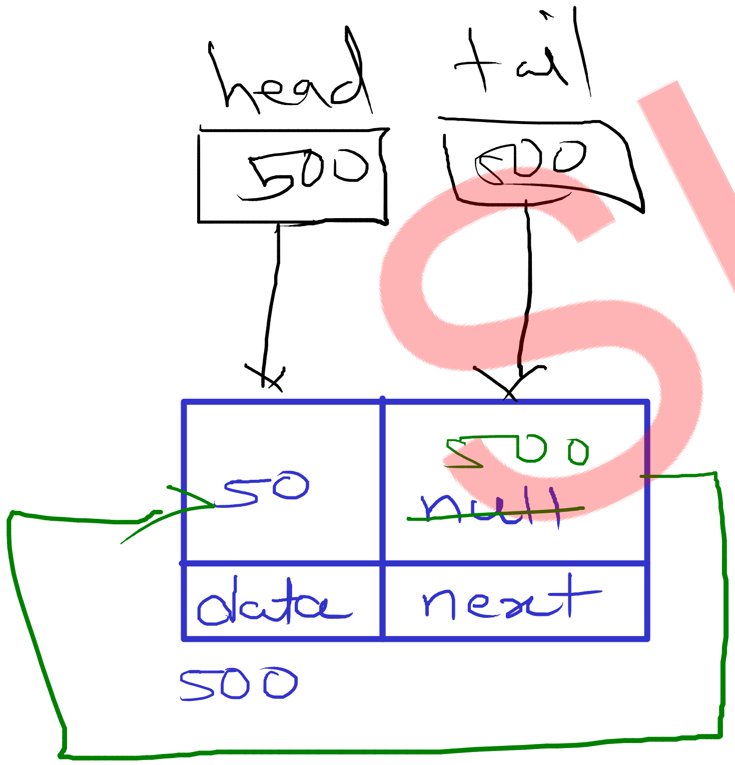
//1. create trav and start at head
//2. visit/print data of current node (trav.data)
//3. go on next node (trav.next)
//4. repeat step 2 and 3 till last node of list

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

$T(n) = O(n)$

Singly Circular Linked List - Add First

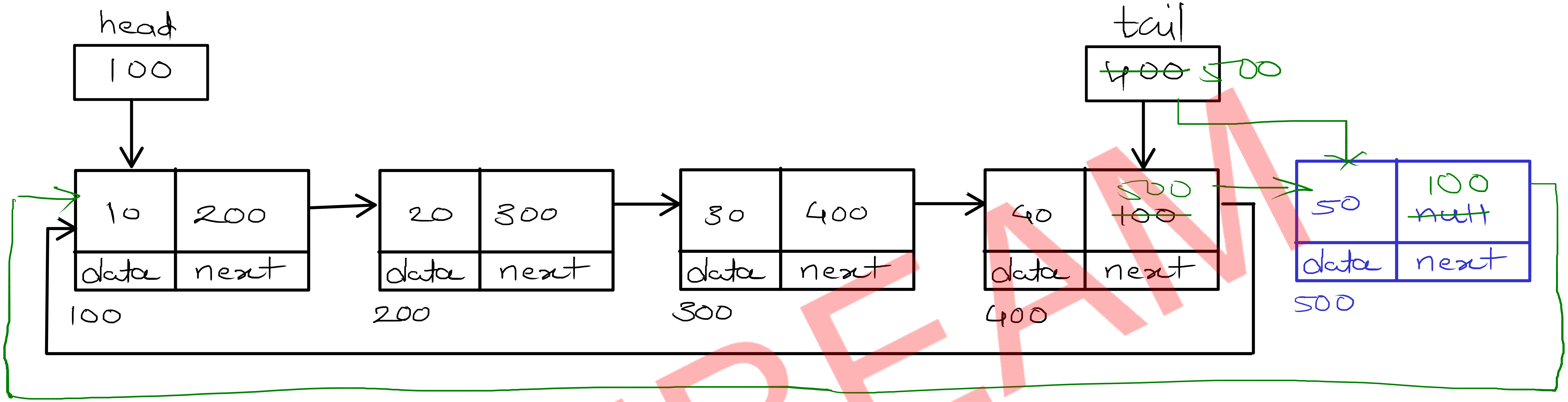


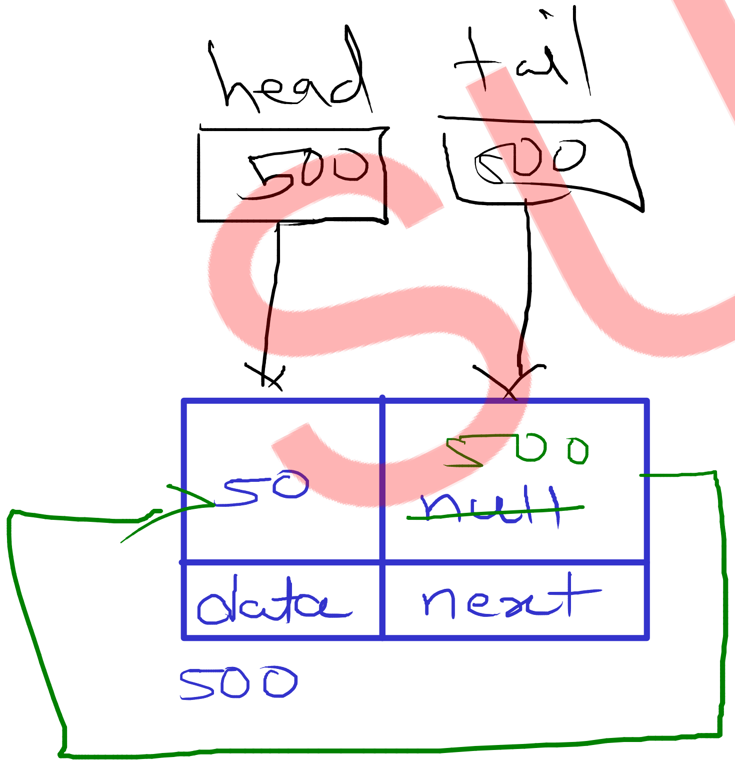


```
//1. create node
//2. if empty
        //a. add newnode into head and tail
        //b. make list circular
//3. if not empty
        //a. add first node into next of newnode
        //b. add newnode into next of last node
        //c. move head on newnode
```

$T(n) = O(1)$

Singly Circular Linked List - Add Last

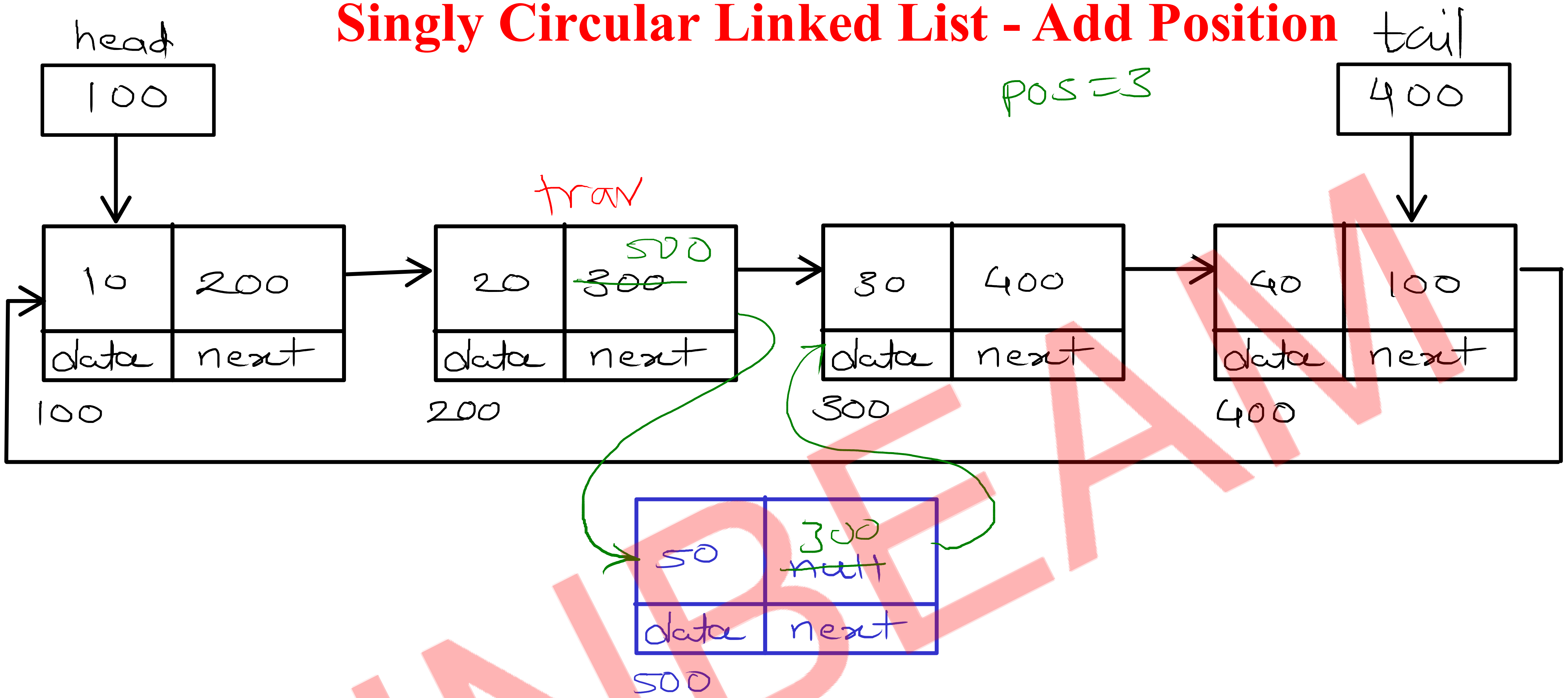




```
//1. create node
//2. if empty
        //a. add newnode into head and tail
        //b. make list circular
//3. if not empty
        //a. add first node into next of newnode
        //b. add newnode into next of last node
        //c. move tail on newnode
```

$T(n) = O(1)$

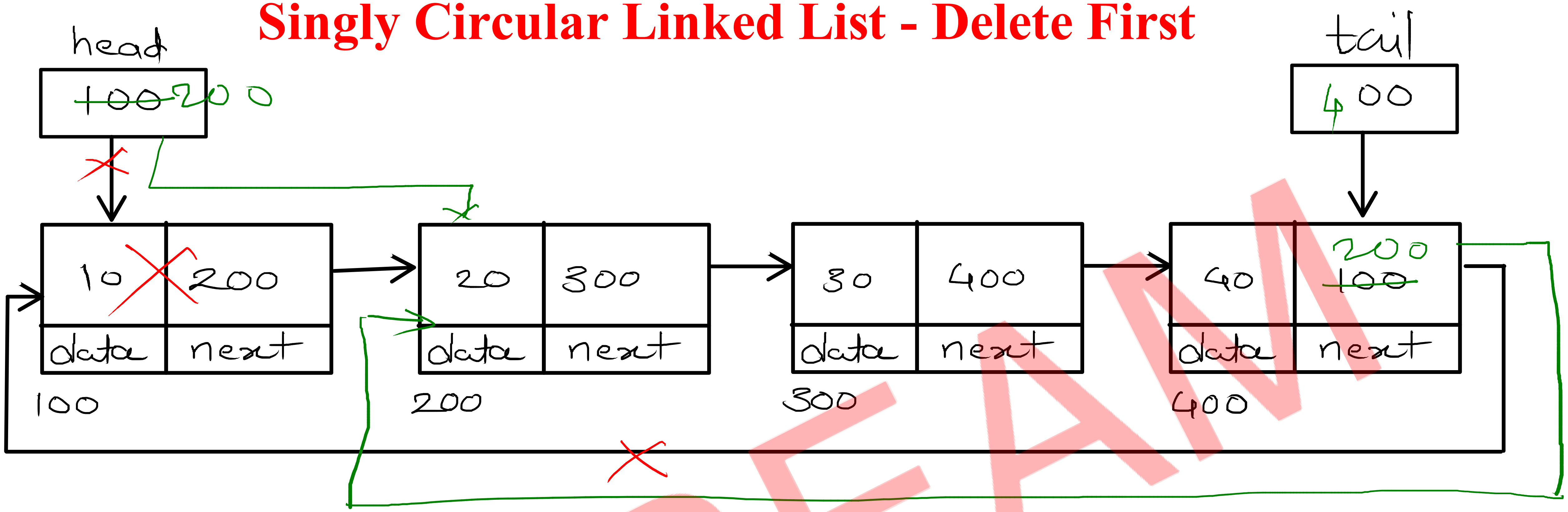
Singly Circular Linked List - Add Position

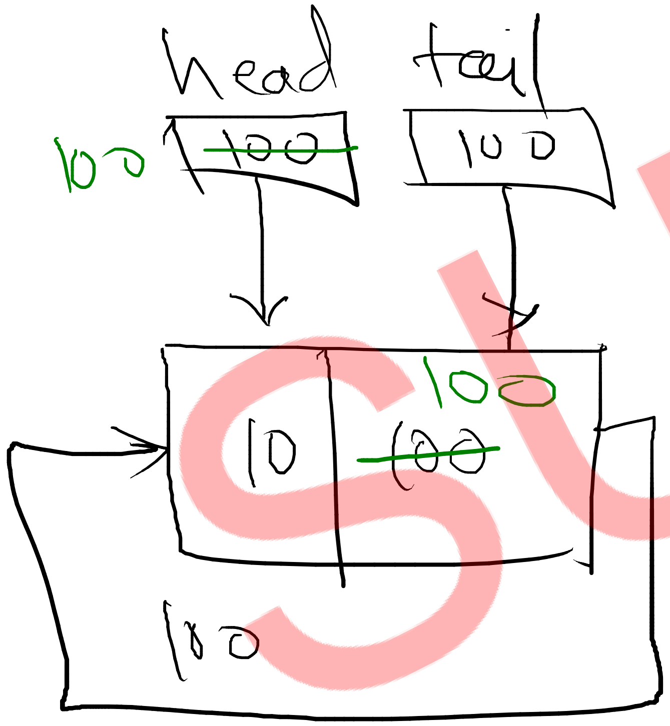


```
//1. create newnode with given data
//2. if list is empty
        // add newnode into head and tail itself
//3. if list is not empty
        //a. traverse till pos -1 node
        //b. add pos node into next of newode
        //c. add newnode into next of pos -1 node
```

$T(n) = O(n)$

Singly Circular Linked List - Delete First



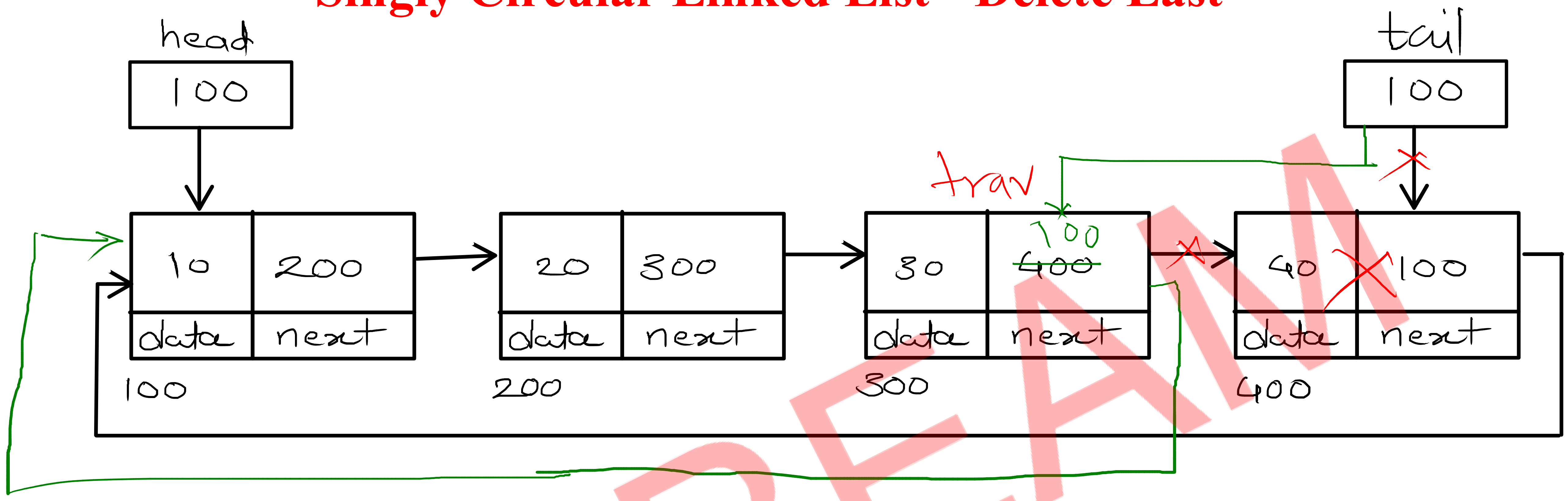


```
//1. if empty
        return;
//2. if single node
        // add null into head and tail
//3. if multiple nodes
        //a. add second node into next of last node
        //b. move head on second node
```

tail.next = head.next
head = head.next

$T(n) = O(1)$

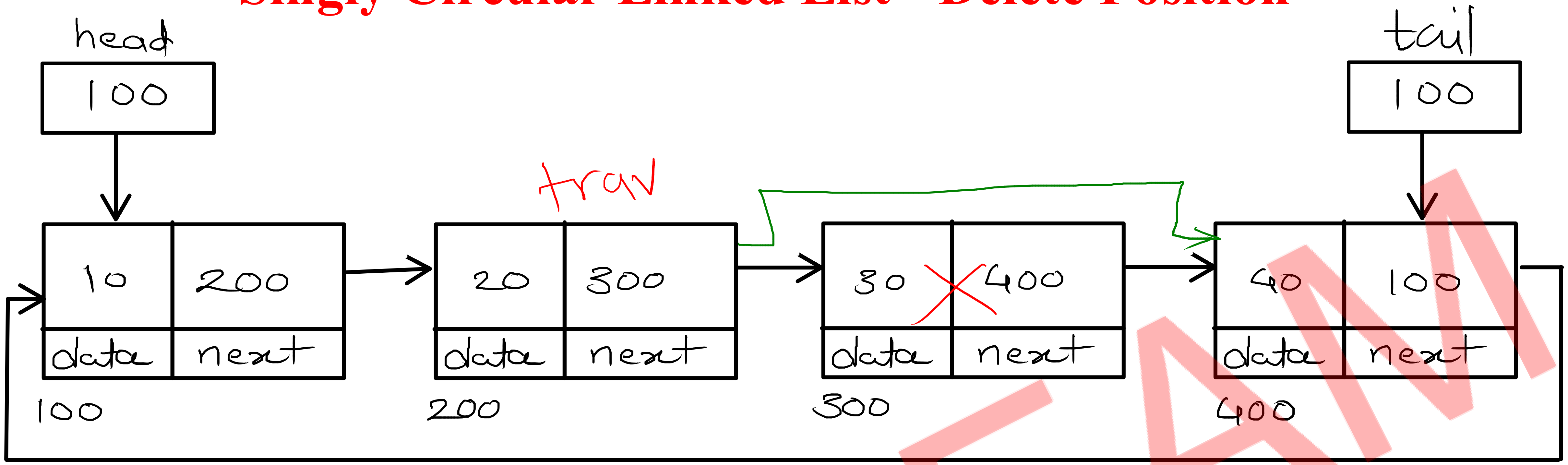
Singly Circular Linked List - Delete Last



```
//1. if empty
        return;
//2. if single node
        // make head and tail equal to null
//3. if multiple nodes
        //a. traverse till second last node
        //b. add first node into next of second last node
        //c. move tail on second last node
```

$T(n) = O(n)$

Singly Circular 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 -1 node
        //b. add pos+1 node into next of pos-1 node
```

T(n) = O(n)