Sunbeam Institute of Information Technology
Pune and Karad

Data structures and Algorithms

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

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

$\Rightarrow$ trav = null

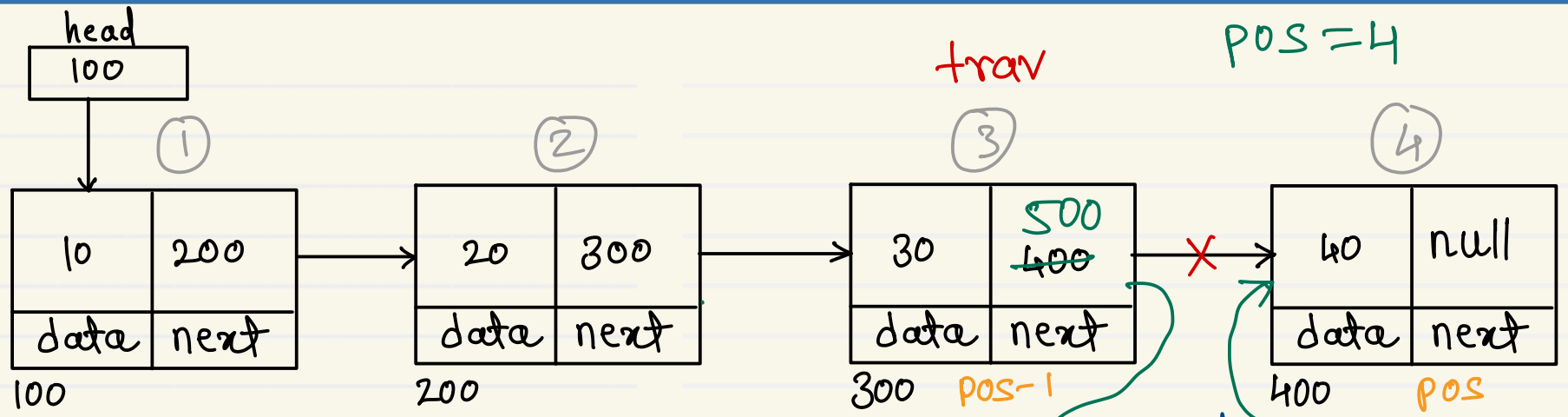
```
trav = head;
while( trav.next != null)
    trav = trav.next;
```

=> trav = last node

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

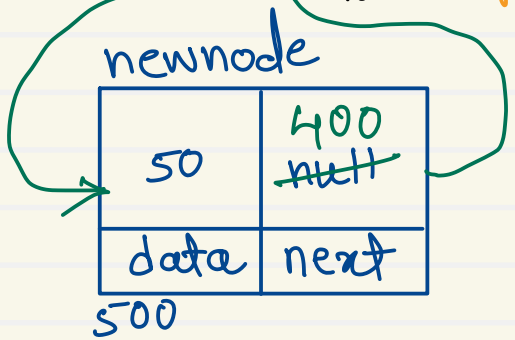
=> trav = second last node

Singly linear Linked List - Add position



1. Create a newnode
2. if list is empty,
   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

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



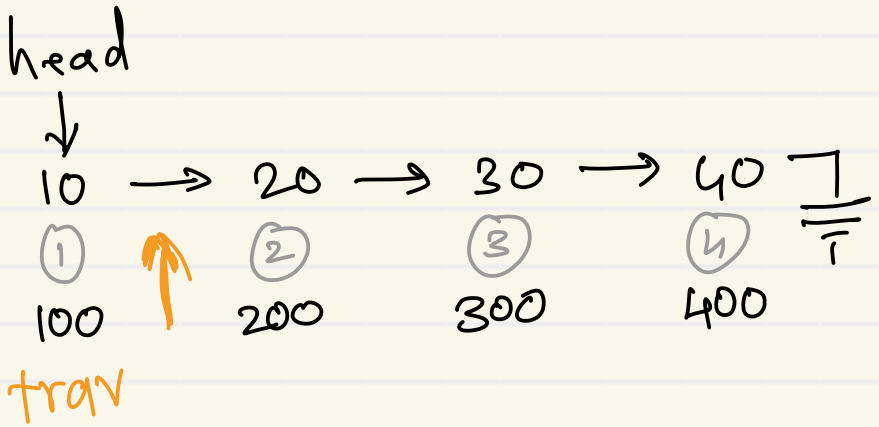
pos=4

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

pos=5

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

$T(n) = O(n)$



pos = 1

trav	i	i < 0
100	1	F

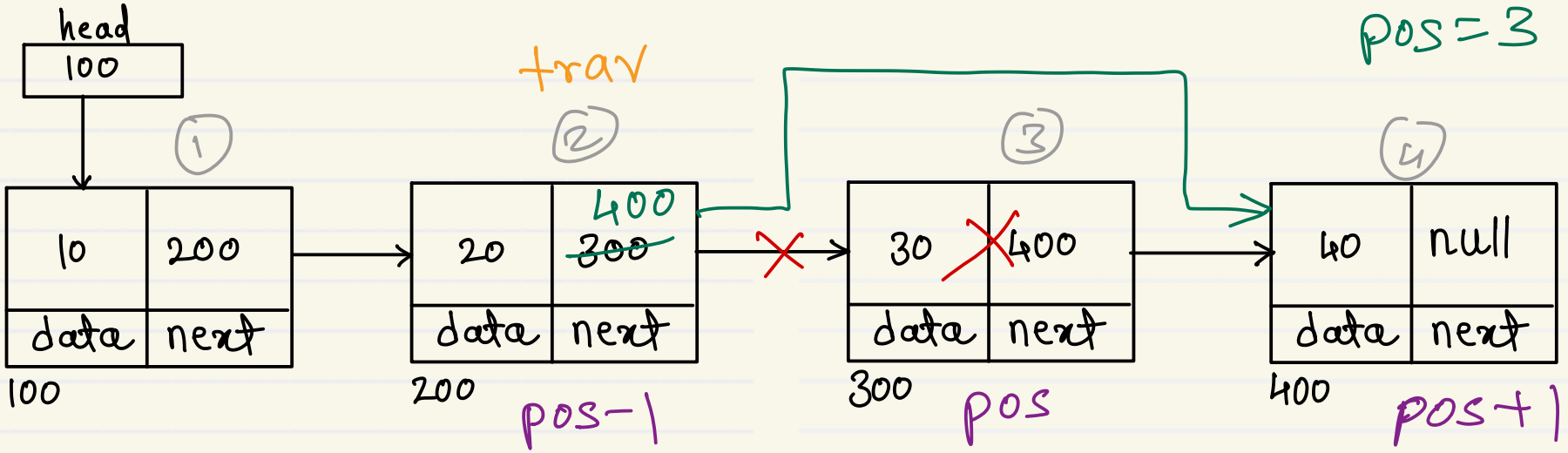
```
trav = head;
for( i=1; i < pos-1 ; i++)
    trav = trav.next;

newnode.next = trav.next
trav.next = newnode
```

pos = 6

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

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

$T(n) = O(n)$

pos = 5

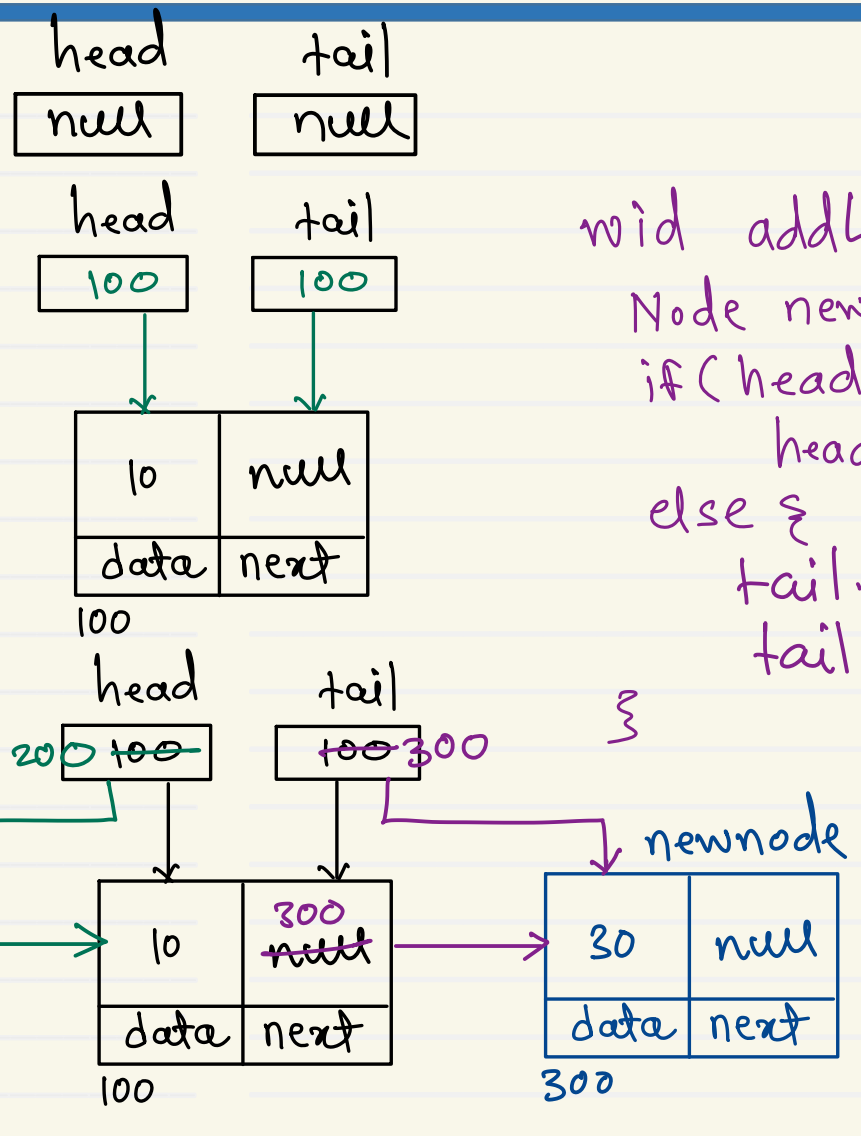
trav	i	i<4
100	1	T
200	2	T
300	3	T
400	4	F

trav.next = trav.next.next
null

Singly linear linked list with head and tail

```
void addFirst(value) {
    Node newnode = new Node(value)
    if(head == null)
        head = tail = newnode;
    else {
        newnode.next = head;
        head = newnode;
    }
```

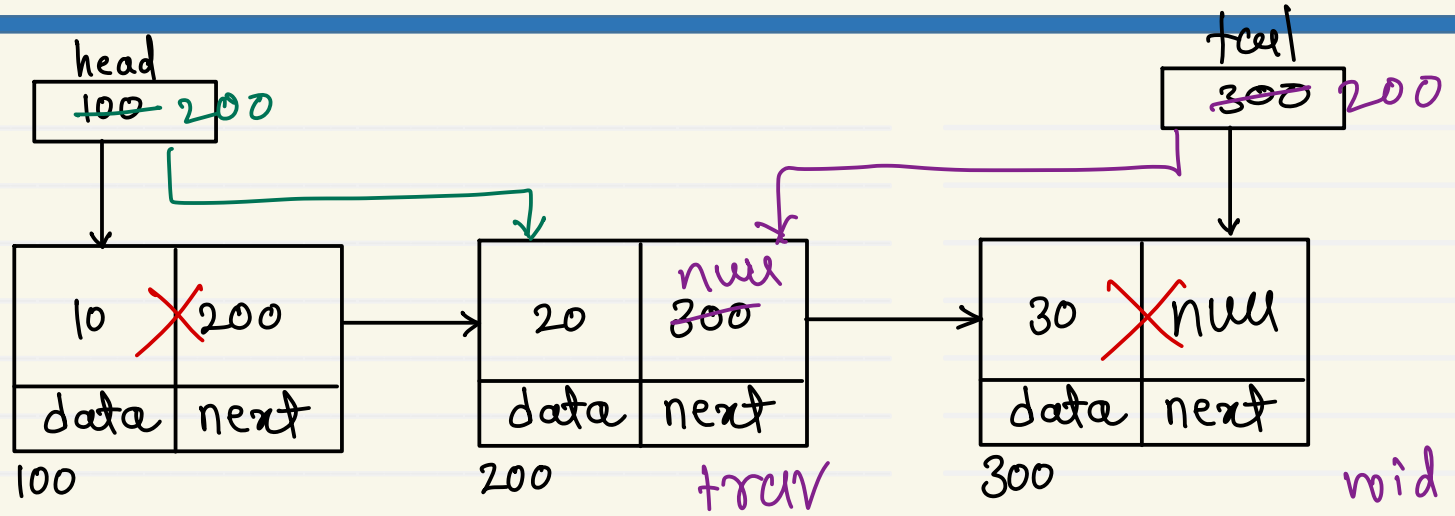
$T(n) = O(1)$




```
void addLast(value) {
    Node newnode = new Node(value)
    if(head == null)
        head = tail = newnode;
    else {
        tail.next = newnode;
        tail = newnode;
    }
}
```

$T(n) = O(1)$

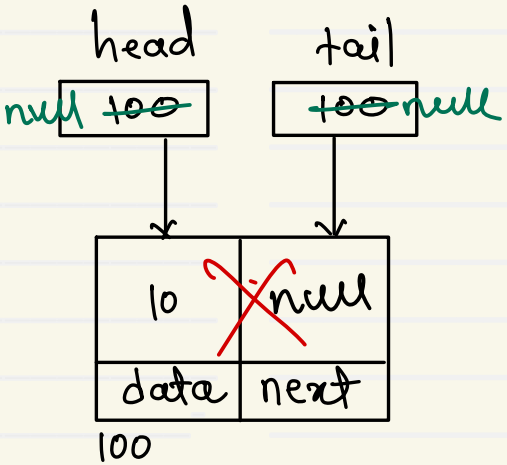
Singly linear linked list with head and tail



$T(n) = O(n)$

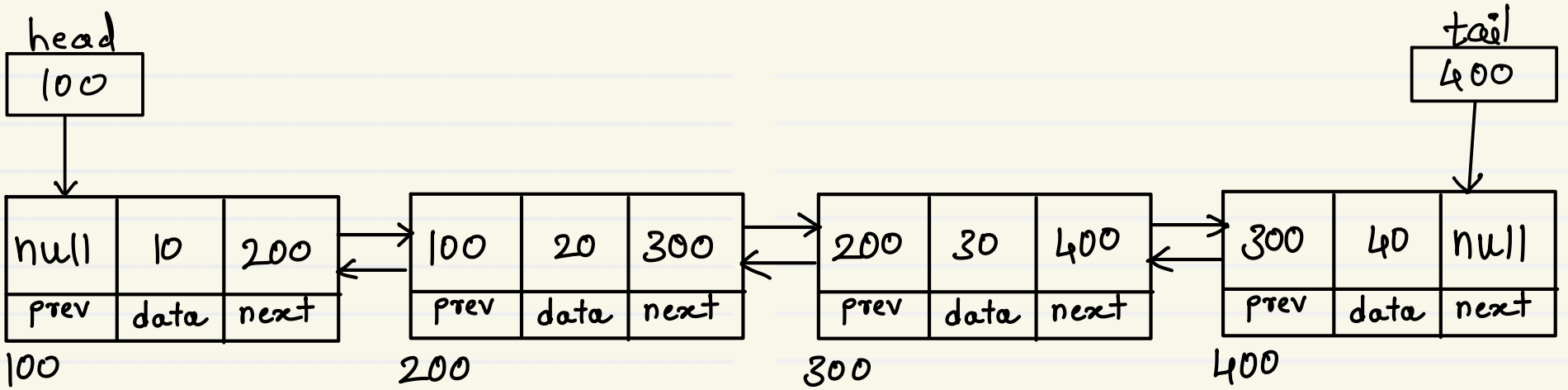
```
void deleteFirst ( ) {
    if ( head == null)
        return;
    else if( head == tail)
        head = tail = null;
    else
        head = head.next;
}
```

$T(n) = O(1)$




```
void deleteLast ( ) {
    if ( head == null)
        return;
    else if( head == tail)
        head = tail = null;
    else {
        Node trav= head;
        while (trav.next.next !=null)
            trav =trav.next;
        tail = trav;
        tail.next=null;
    }
}
```

Doubly Linear Linked List - Display

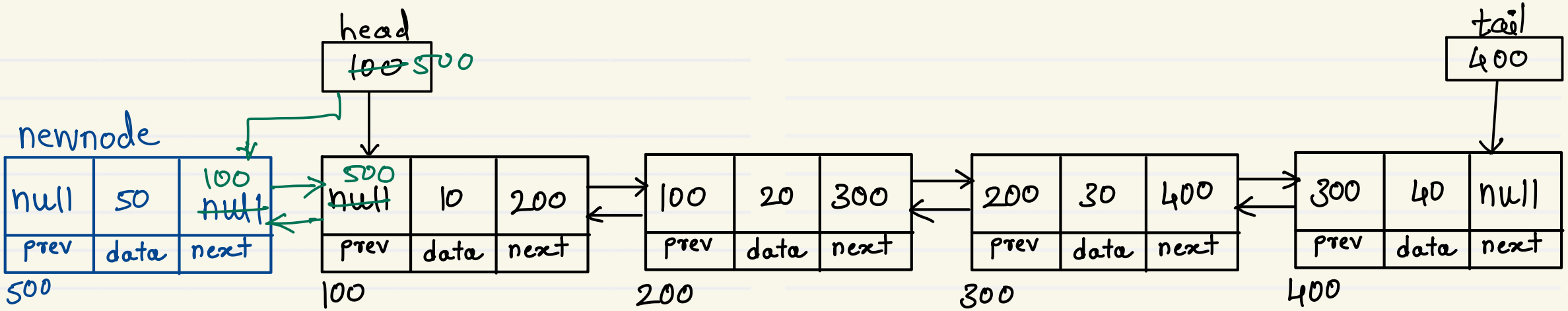



```
trav = head;
while(trav != null) {
    sysout(trav.data);
    trav = trav.next;
}
```

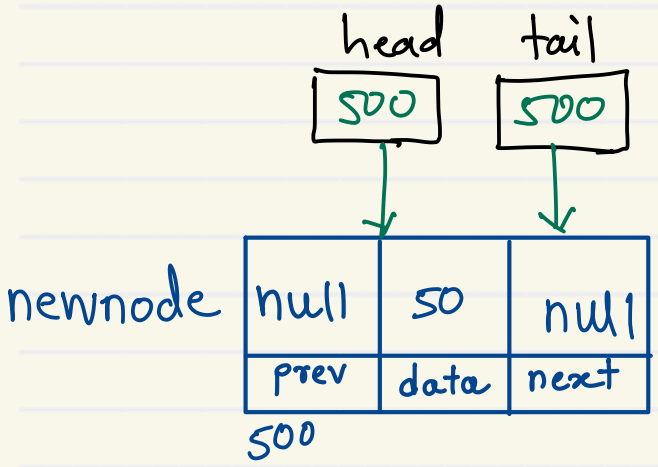
```
trav = tail;
while(trav != null) {
    sysout(trav.data);
    trav = trav.prev;
}
```

$T(n) = O(n)$

Doubly Linear Linked List - Add first

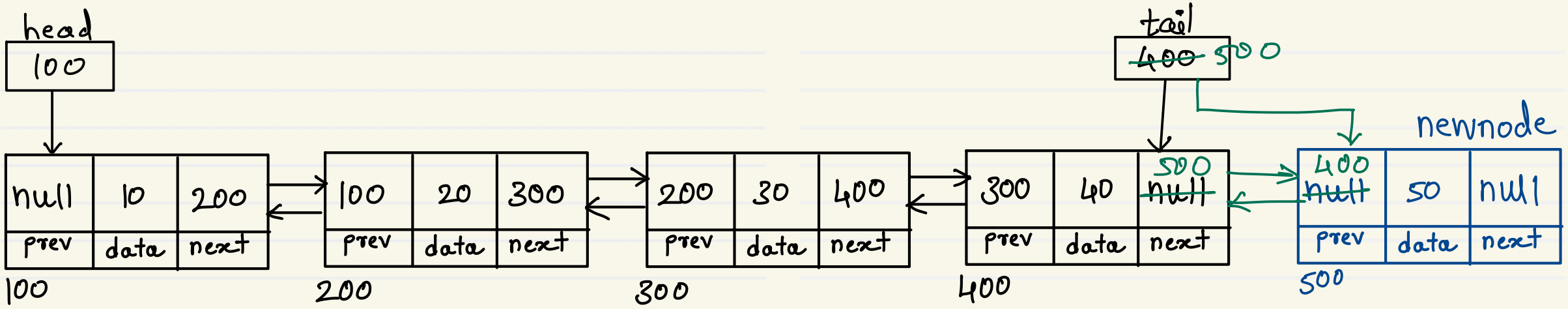


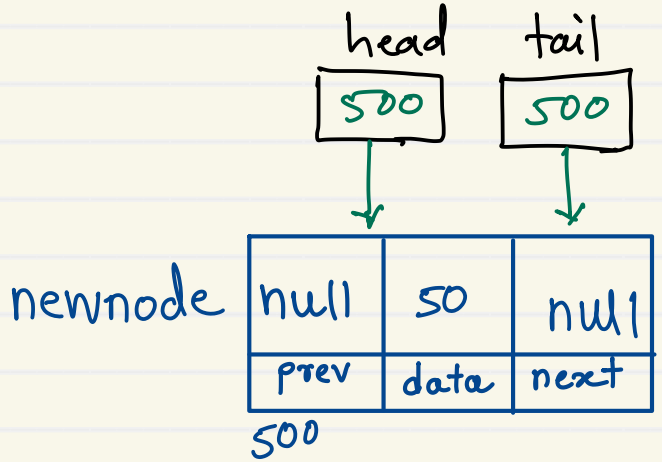
1. Create newnode
2. if list is empty,
   add newnode into head & tail
3. if list is not empty,
   a. add first node into next of newnode
   b. add newnode into prev of first node
   c. move head on newnode



$T(n) = O(1)$

Doubly Linear Linked List - Add last

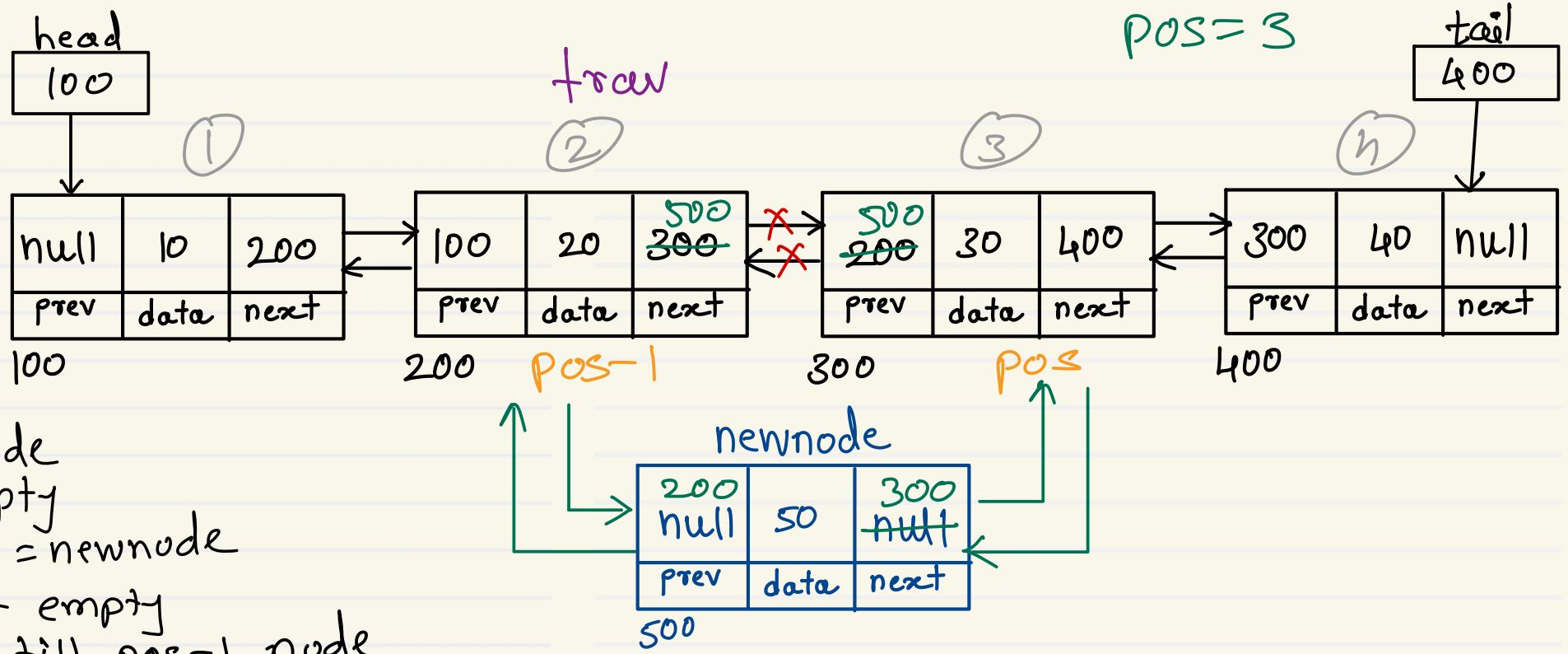




1. Create newnode
2. if list is empty,
   add newnode into head & tail
3. if list is not empty,
   a. add last node into prev of newnode
   b. add newnode into next of last node
   c. move tail on newnode

$T(n) = O(1)$

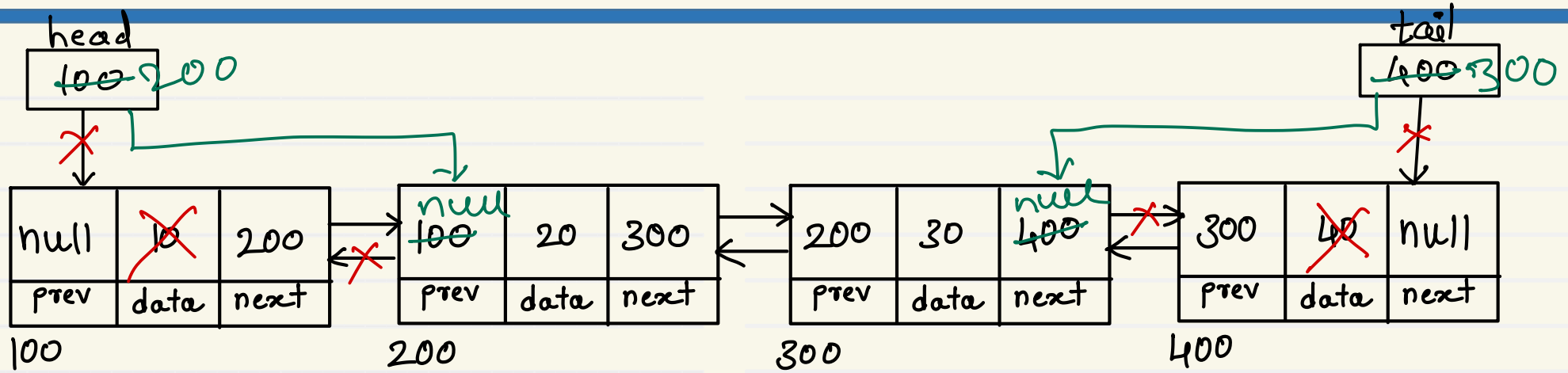
Doubly Linear Linked List - Add position

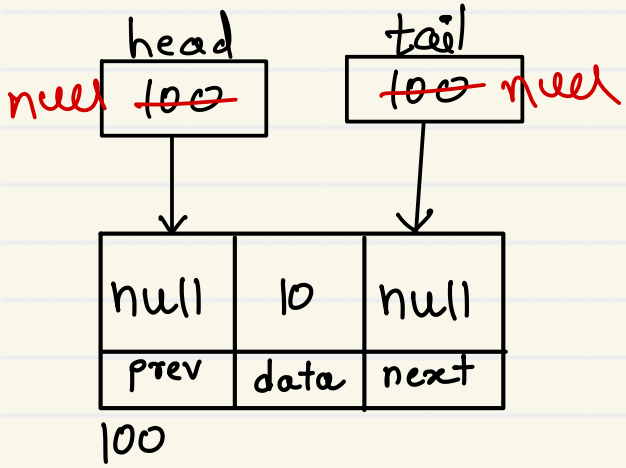


1. create newnode
2. if list is empty
   head = tail = newnode
3. if list is not empty
   a. traverse till pos-1 node
   b. add pos-1 node into prev of newnode
   c. add pos node into next of newnode
   d. add newnode into next of pos-1 node
   e. add newnode into prev of pos node

$$T(n) = O(n)$$

Doubly Linear Linked List - Delete first and last





delete first :

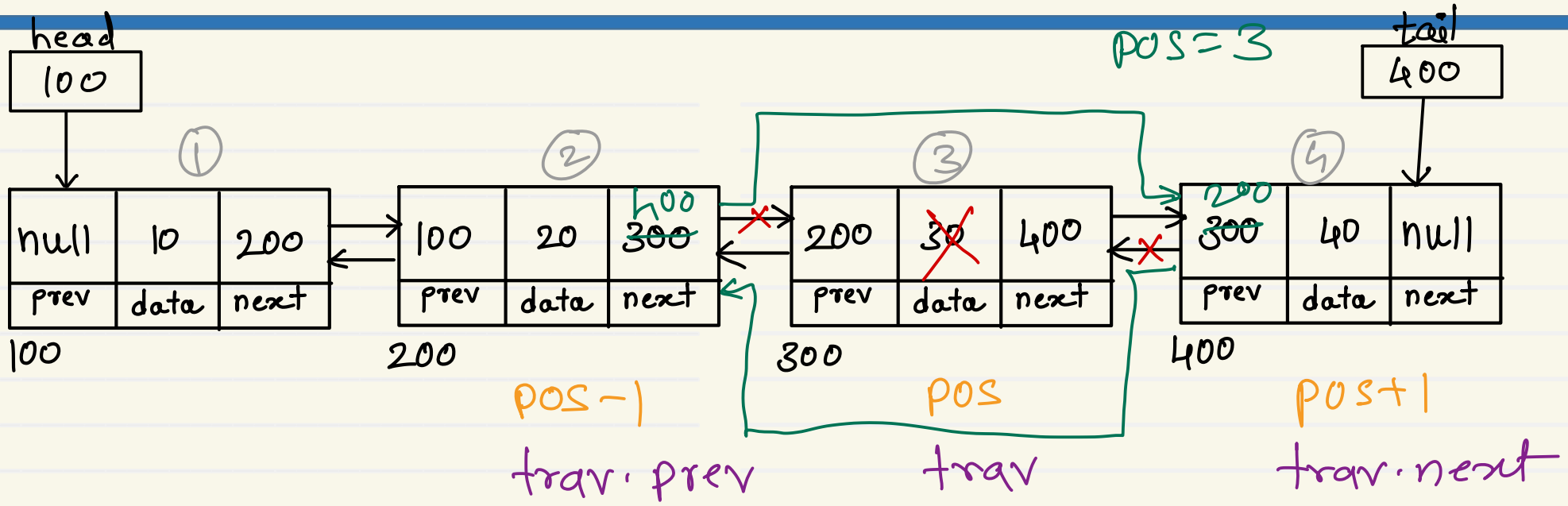
```
head = head.next;
head.prev = null;
```

delete last :

```
tail = tail.prev
tail.next = null;
```

T(n) = O(1)

Doubly Linear Linked List - Delete Position



if list is not empty
- a. add pos+1 node into next of pos-1 node
- b. add pos-1 node into prev of pos+1 node

$T(n) = O(n)$