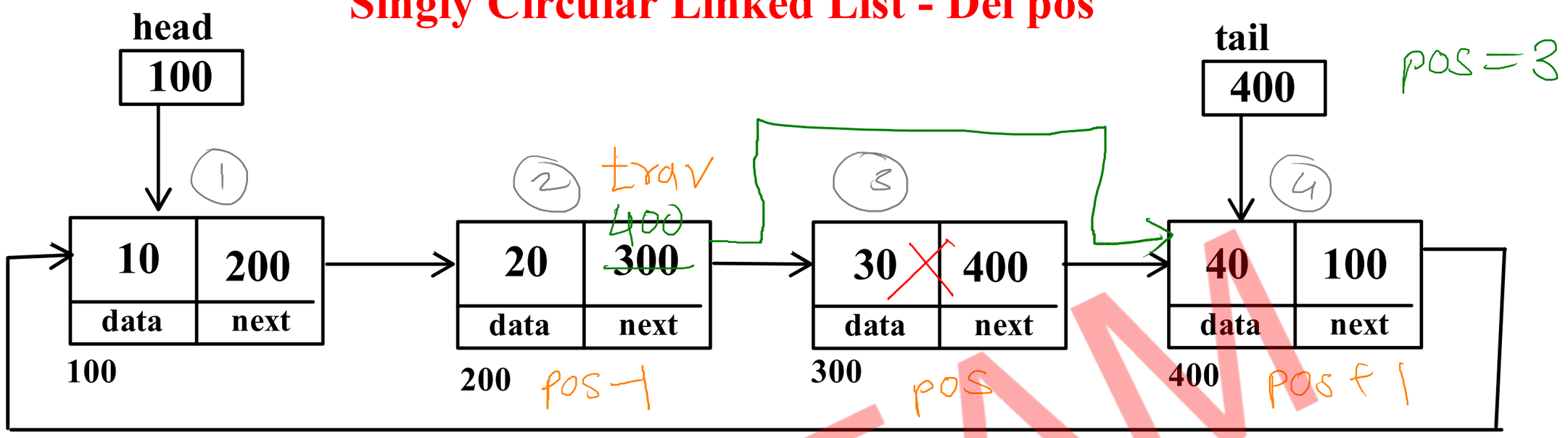
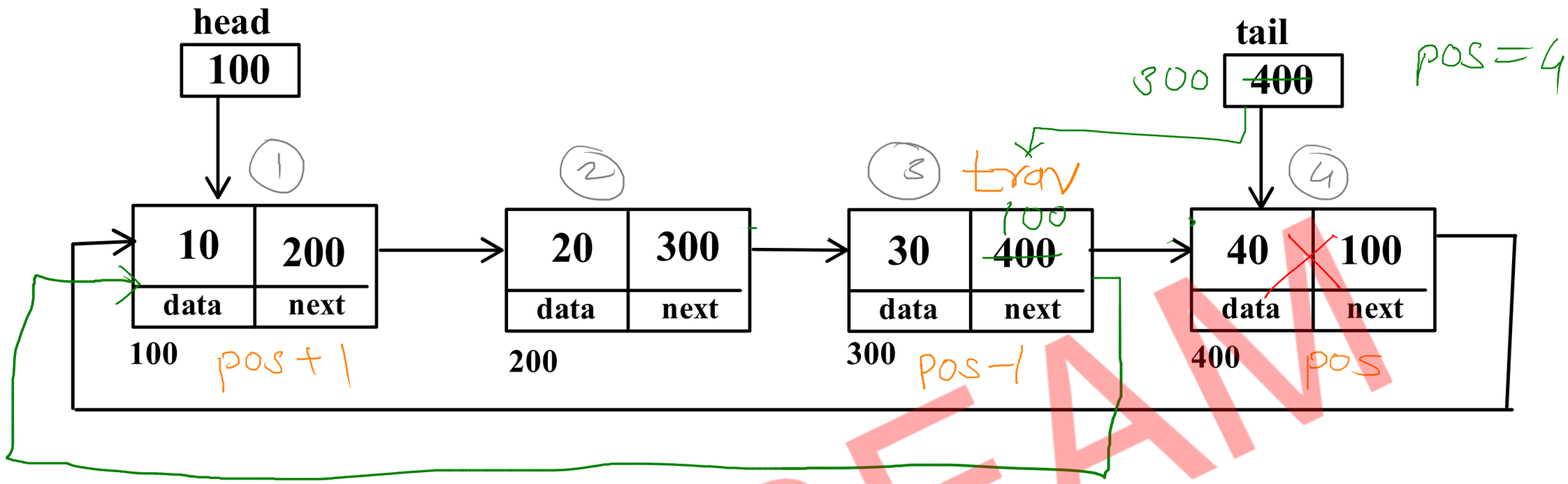
Singly Circular Linked List - Del pos




```
//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)

SUNBEAM



//3. if list has multiple nodes

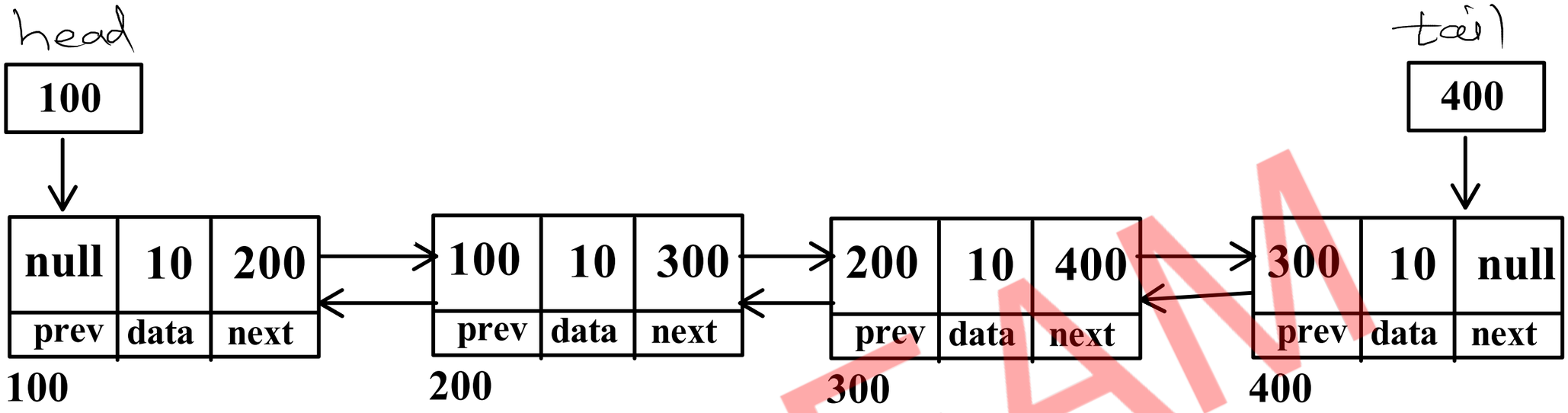
//a. traverse till pos-1 node

// special case : if pos is last node of list, move tail on second last node

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

SUNBEAM

Doubly Linear Linked List - Display



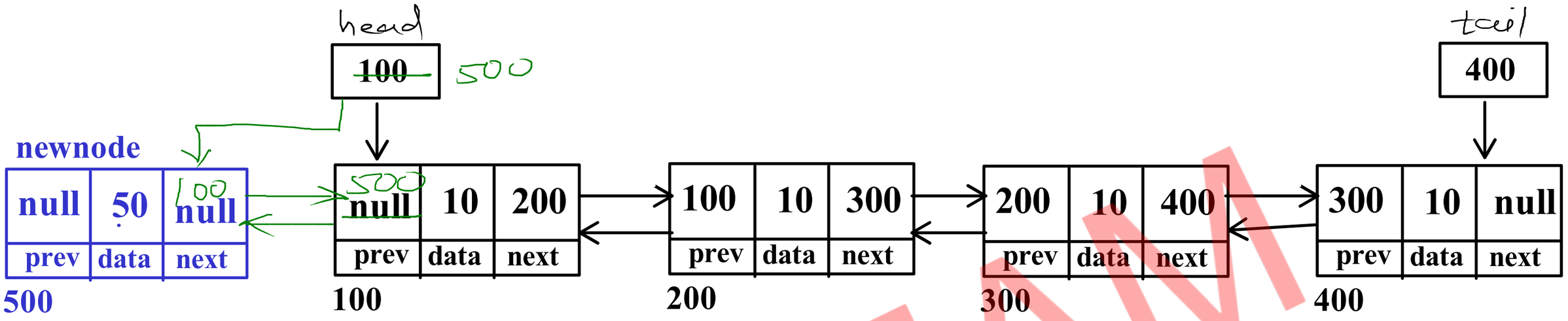

```
// forward display
//1. create trav and start at first node
//2. print/visit current node(trav.data)
//3. go on next node(trav.next)
//4. repeat step 2 and 3 till last node
```

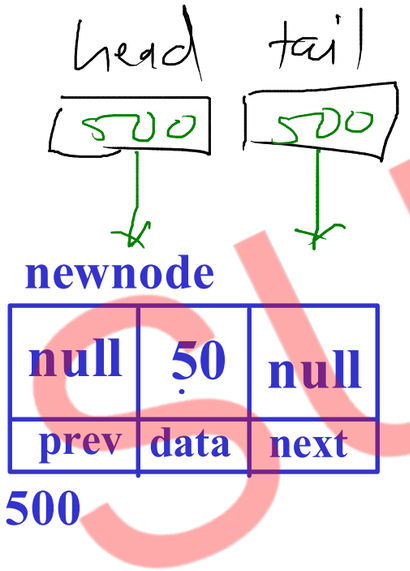
```
// reverse display
//1. create trav and start at last node
//2. print/visit current node(trav.data)
//3. go on prev node(trav.prev)
//4. repeat step 2 and 3 till first node
```

$T(n) = O(n)$

SUNBEAM

Doubly Linear Linked List - Add first



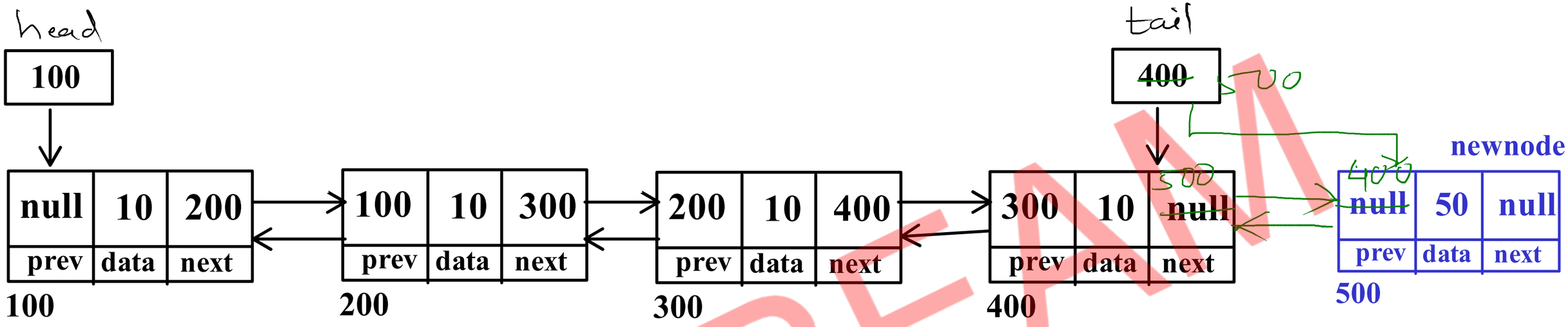


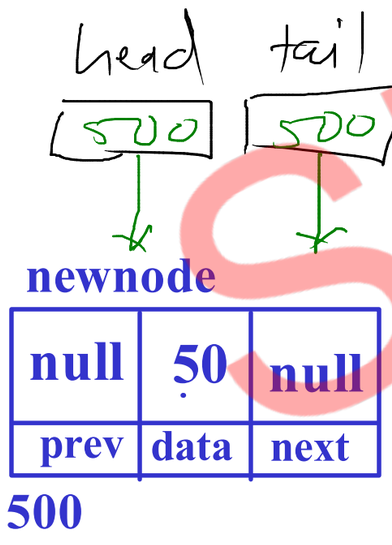

```
//1. create node for data
//2. if list is empty
      // add newnode into head and tail
//3. if list is not empty
      //a. add first node into next of newnode
      //b. add newnode into prev of first node
      //c. move head on newnode
```

T(n) = O(1)

SUNBEAM

Doubly Linear Linked List - Add Last



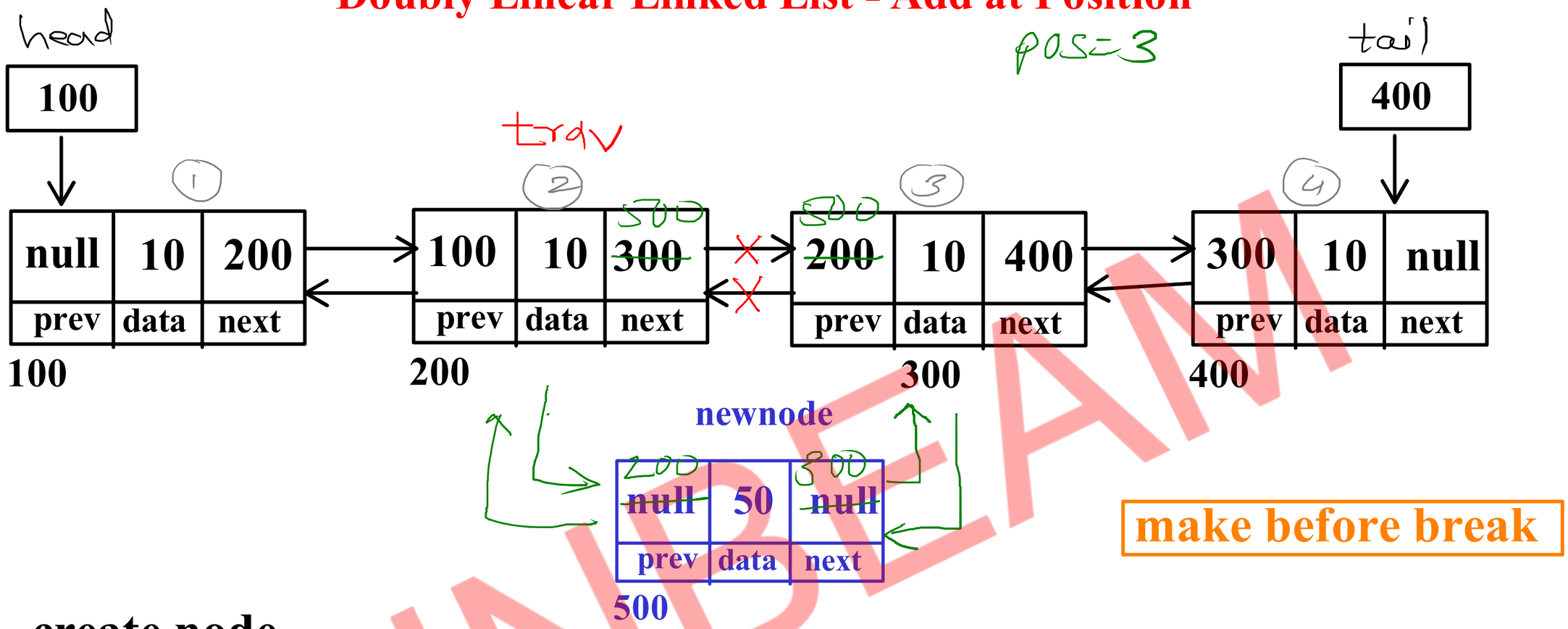



```
//1. create node for data
//2. if list is empty
        // add newnode into head and tail
//3. if list is not empty
        //a. add last node into prev of newnode
        //b. add newnode into next iof last node
        //c. move tail on newnode
```

T(n) = O(1)

SUNBEAM

Doubly Linear Linked List - Add at Position



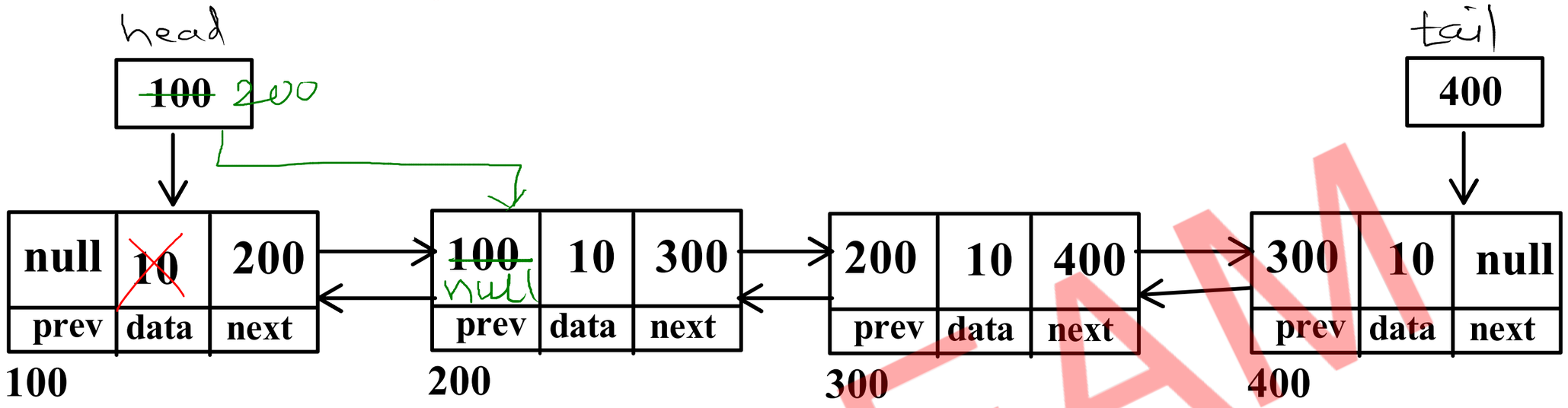
make before break

```
//1. create node
//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)

SUNBEAM

Doubly Linear Linked List - Delete First

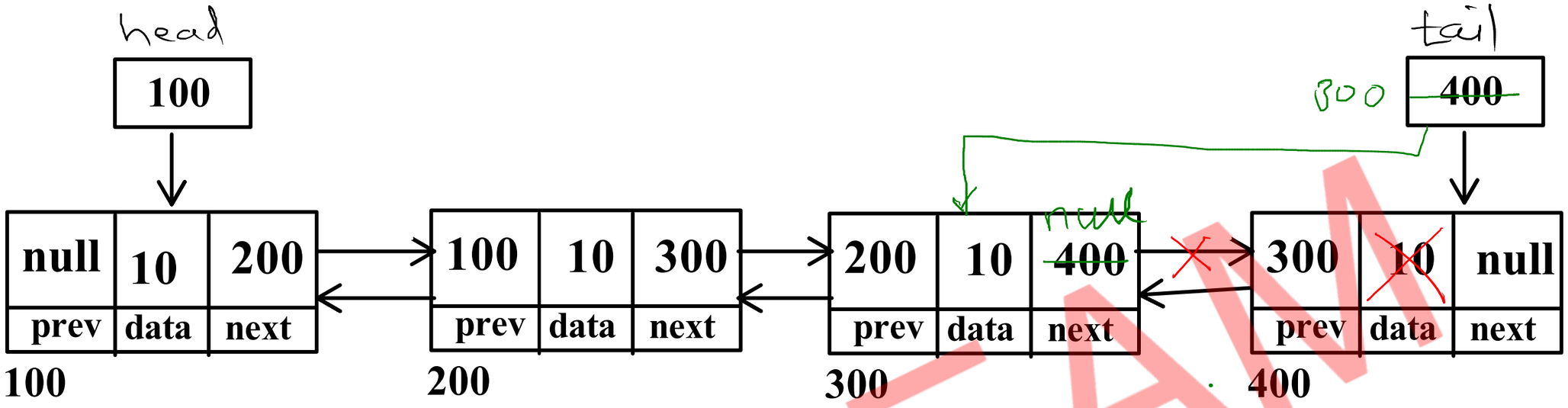



```
//1. if list is empty
        return;
//2. if list has single node
        head = tail = null;
//3. if list has multiple nodes
        //a. move head on second node
        //b. make prev of second node equal to null
```

T(n) = O(1)

SUNBEAM

Doubly Linear Linked List - Delete Last

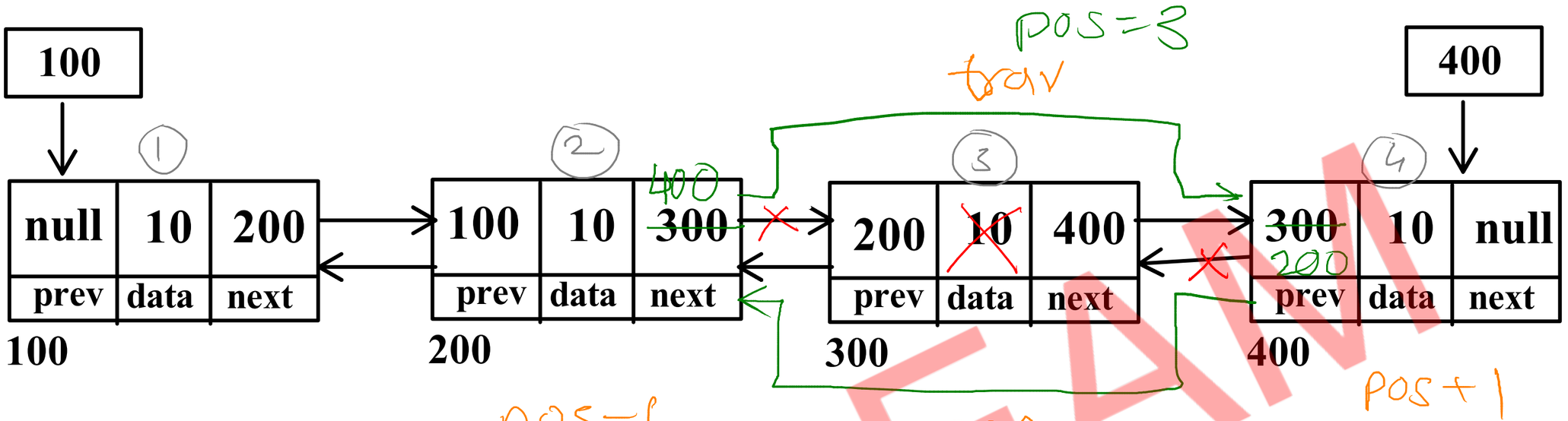



```
//1. if list is empty
     return;
//2. if list has single node
     head = tail = null;
//3. if list has multiple nodes
     //a. move tail on second last node
     //b. make next of second last node equal to null
```

T(n) = O(1)

SUNBEAM

Doubly Linear Linked List - Delete Position

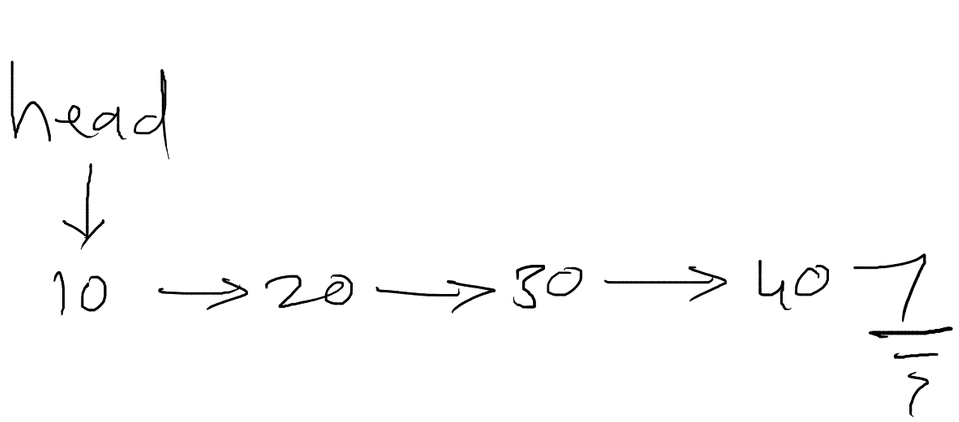


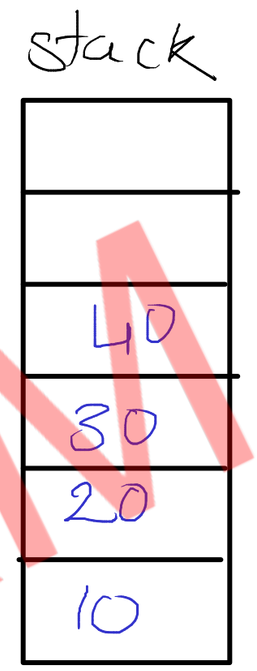

```
//1. if list is empty
        return;
//2. if list has single node
        head = tail = null;
//3. if list has multiple nodes
        //a. traverse till pos node
        //b. add pos+1 node into next of pos-1 node
        //c. add pos-1 node into prev of pos+1 node
```

T(n)=O(n)

SUNBEAM

Display Singly linear linked in reverse order





Output : 40, 30, 20, 10

```
void reverseDisplay() {
    Stack<Integer> st = new stack<>();
    for(Node trav=head; trav!=null; trav=trav.next)
        st.push(trav.data);
    while(!st.isEmpty())
        sysout(st.pop());
}
```

extra space (auxillary space)

← n
+= 2n
← n

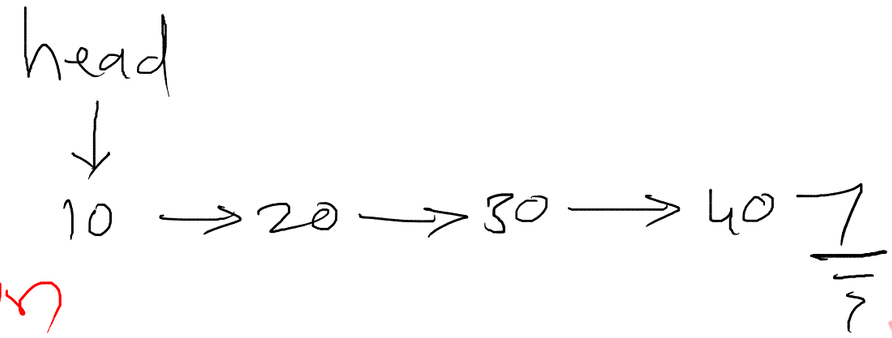
Time ∝ 2n

T(n) = O(n)

AS(n) = O(n)

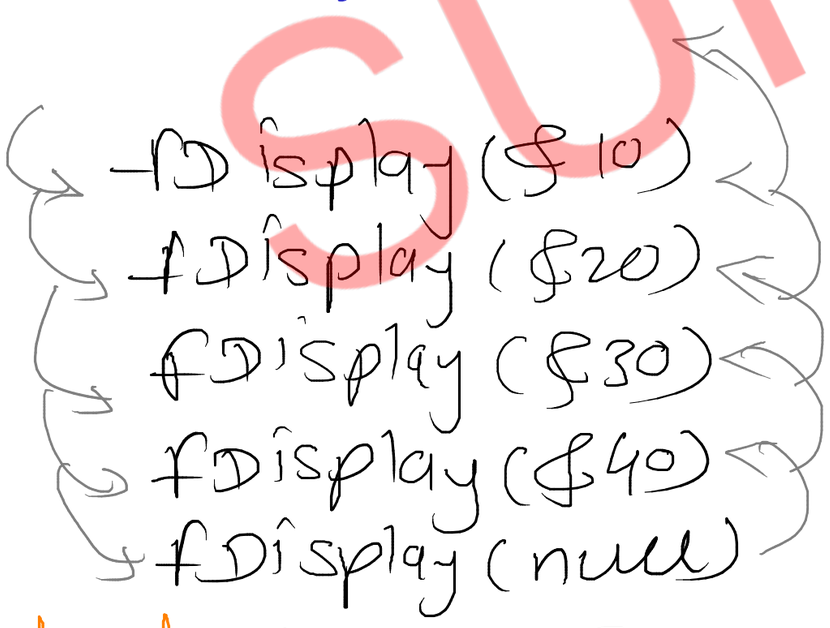
SUNBEAM

Display Singly linear linked in reverse order



Tail Recursion

```
void fDisplay(Node trav)
{
    if(trav == null)
        return;
    sysout(trav.data);
    fDisplay(trav.next);
}
```



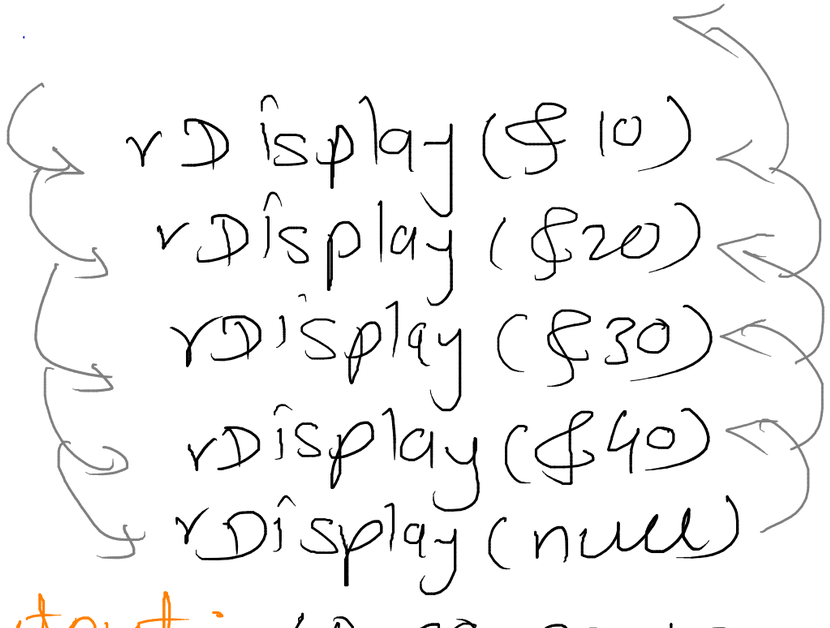
Output : 10, 20, 30, 40

T(n) = O(n)

AS(n) = O(1)

Non tail Recursion

```
void rDisplay(Node trav)
{
    if(trav == null)
        return;
    rDisplay(trav.next);
    sysout(trav.data);
}
```



Output : 40, 30, 20, 10

SUNBEAM