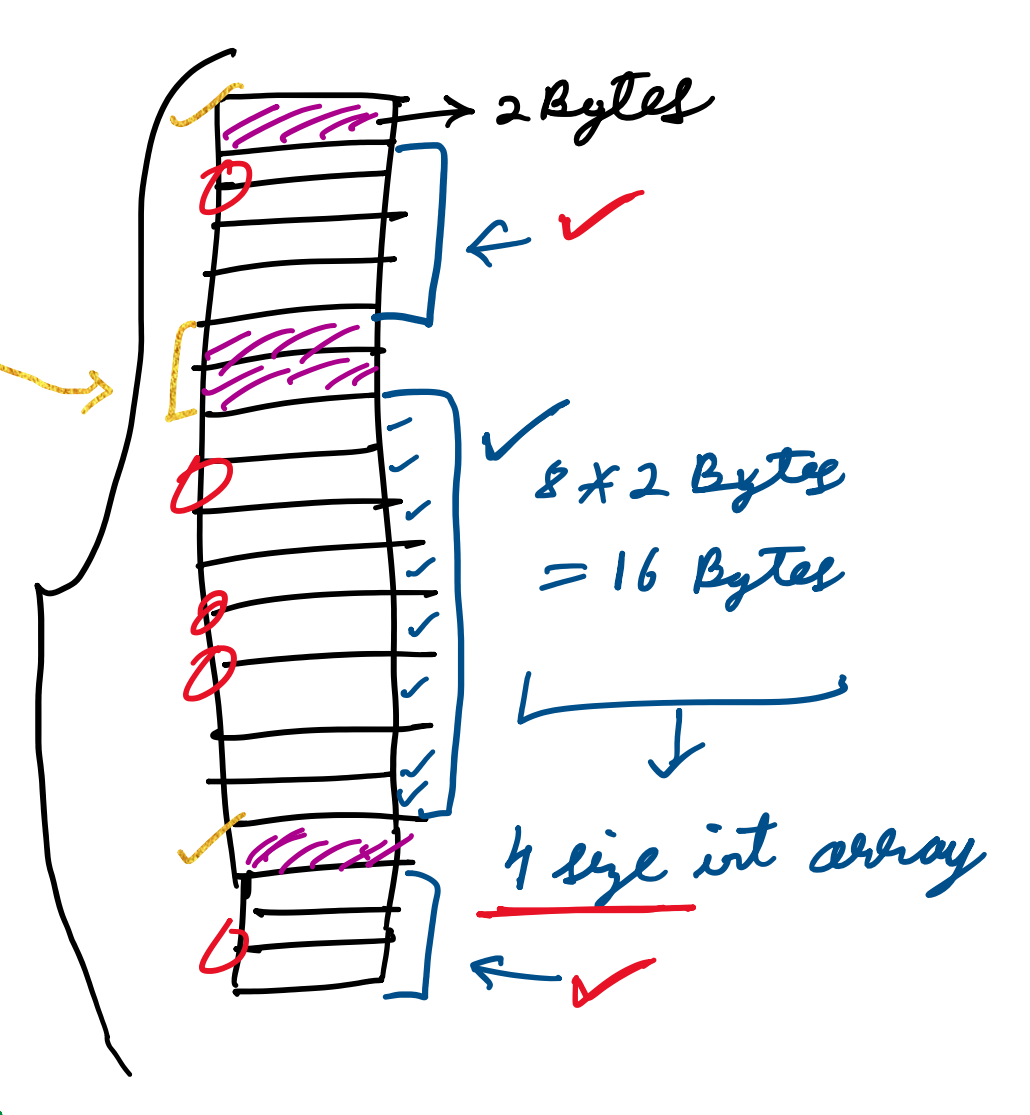
Linked list 1

Monday, 3 January 2022 9:02 PM

int a = 10;

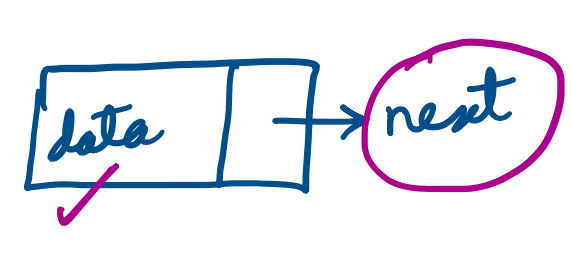
Max size of (int) array?

4 Bytes

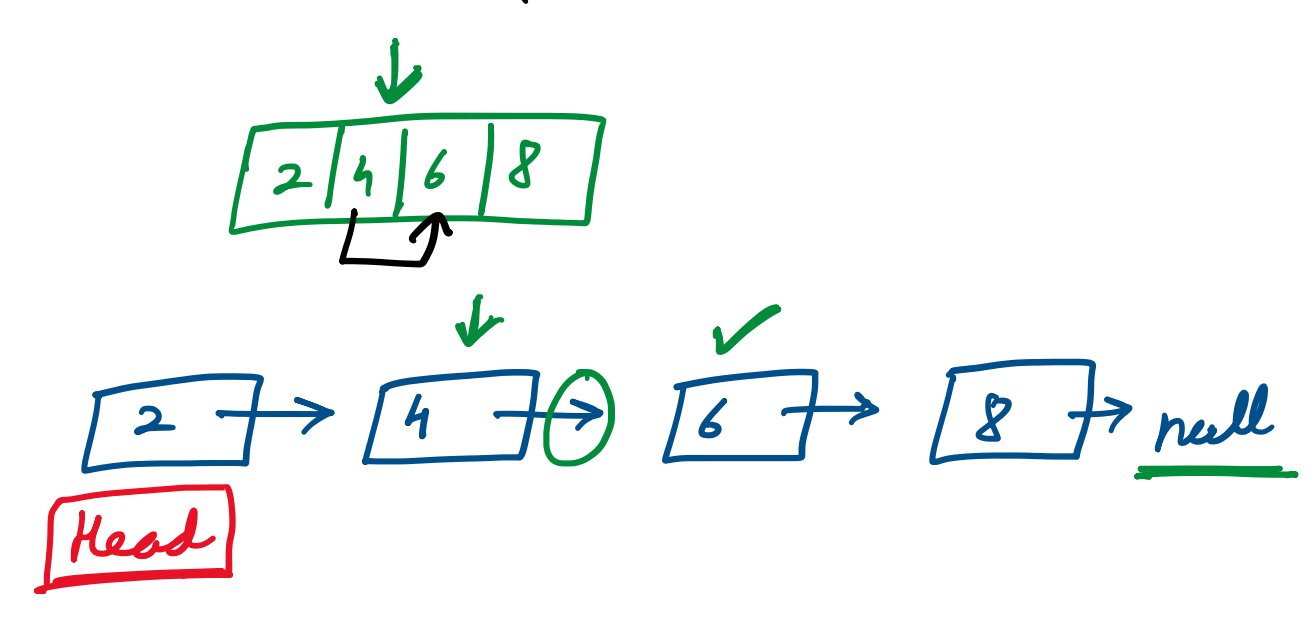


Linked List

1) Linear DS
2) Non contiguous memory allocation ✓




```
struct node {
    int data;
    node * next;
}
```

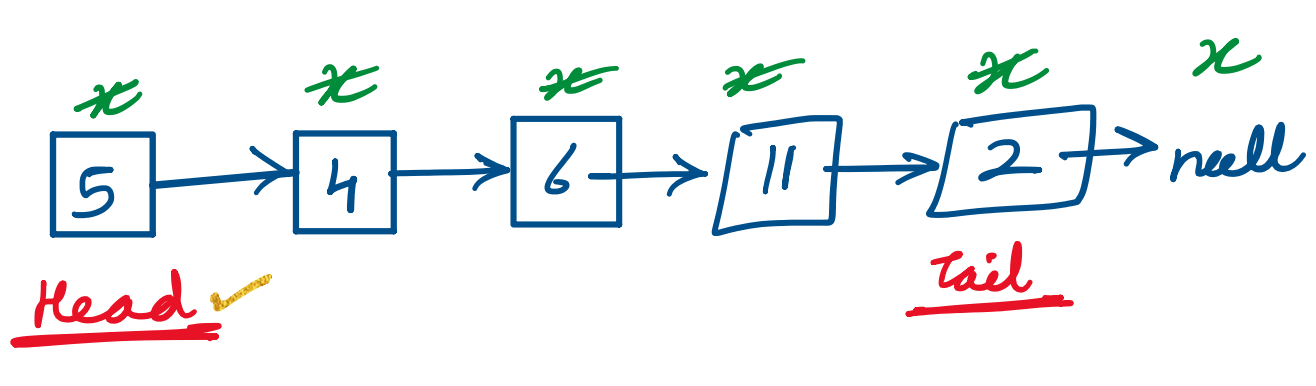


[Always make a copy of head node before doing any operations.]

```
class node {
    int data;
    node next;
    node (int x) {
        data = x;
        next = null;
    }
}
```

Operations on LL

1) Access K^{th} element



Array → A[K-1] TC=O(1) K=6

```
node x = head;
for (i = 1; i < K; i++) {
    if (x == null) return -1;
    x = x.next;
}
return (x == null ? -1 : x.data);
```

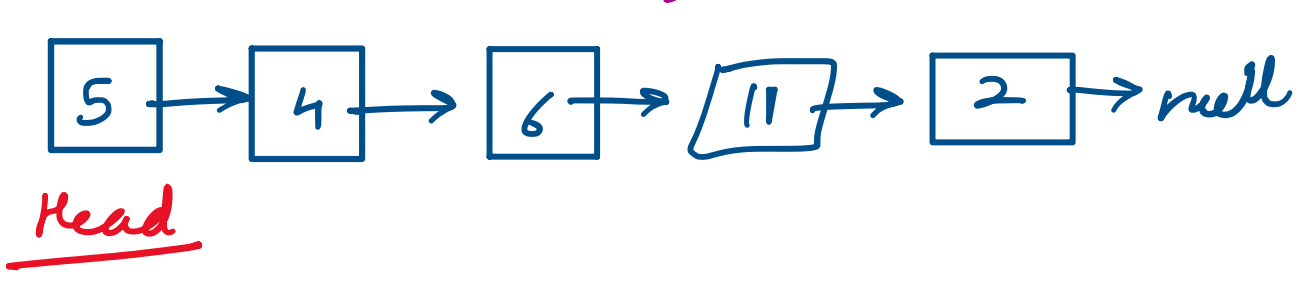
i = 1 2 3 4 5 6

TC=O(K)

2) Search value V

TC in Arrays → O(N) linear search
→ O(log(N)) binary search

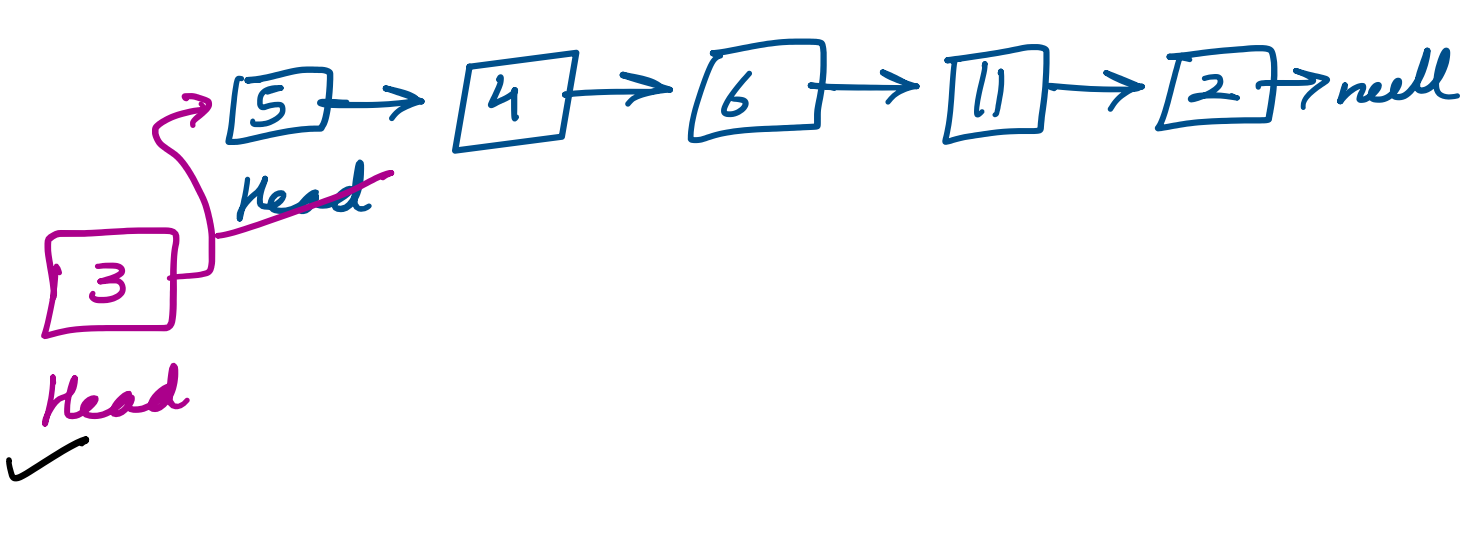
Eg V = 11




```
node x = head;
while (x != null) {
    if (x.data == v) return x;
    x = x.next;
}
return null;
```

3) Insert a value V

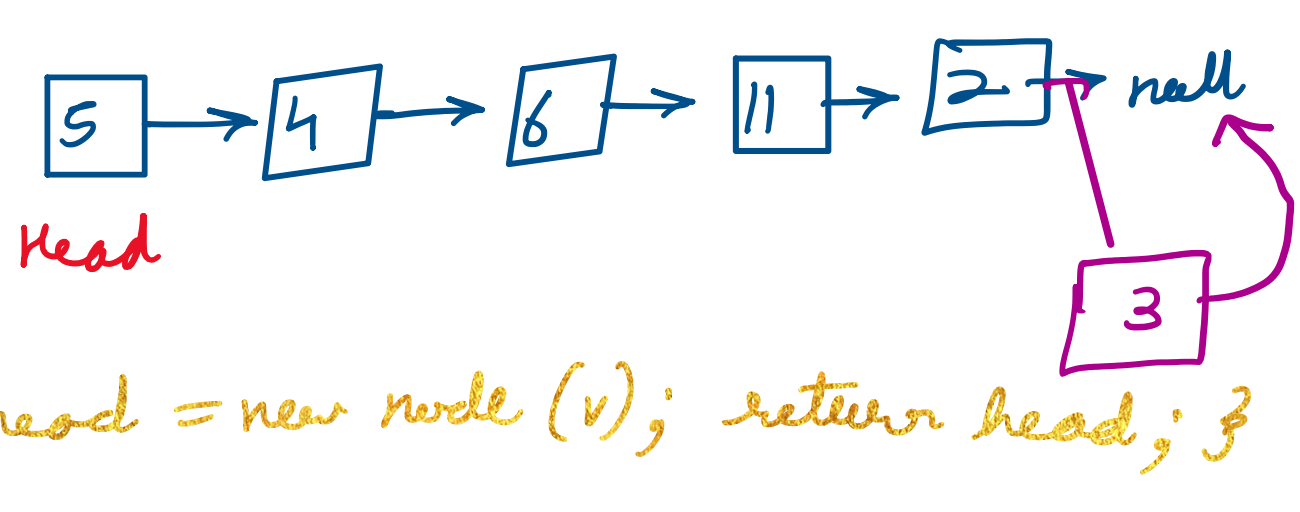
a) At head




```
node x = new node(v);
x.next = head;
head = x;
return head;
```

TC=O(1)

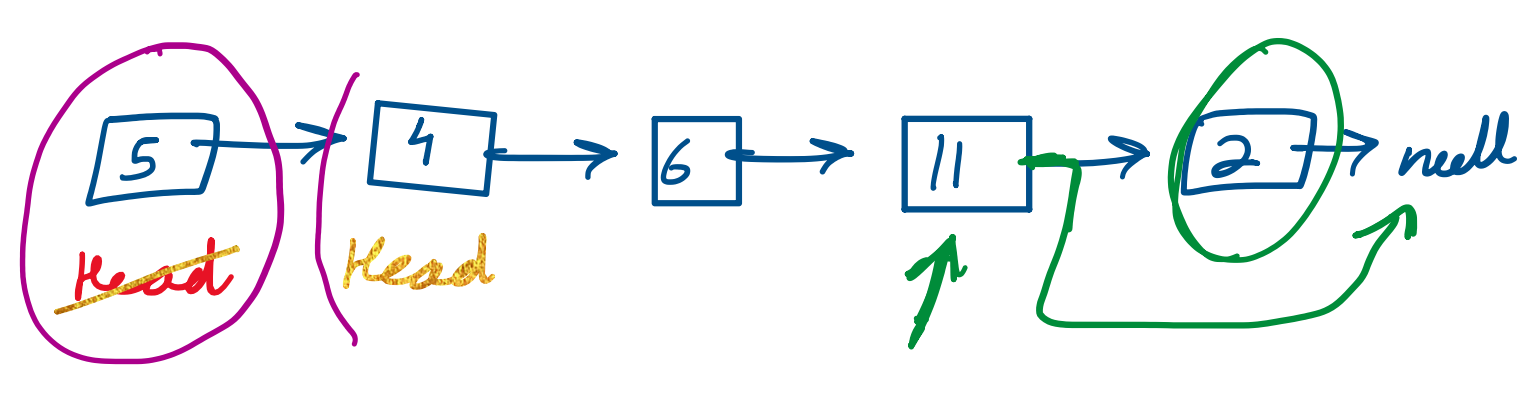
b) At tail




```
if(head == null) { head = new node (v); return head; }
node x = head;
while( x.next != null) {
    x = x.next;
}
x.next = new node (v);
return head;
```

TC=O(N)

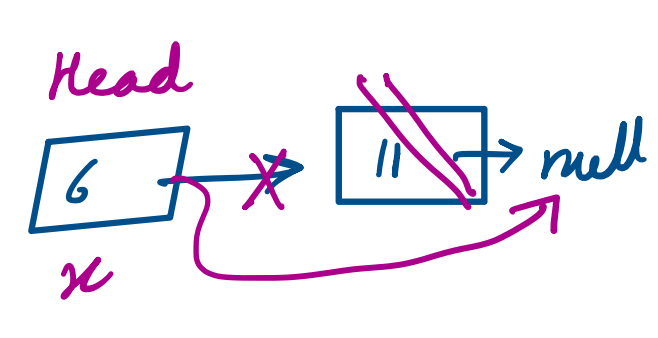
4) Deletion



a) of Head node

```
{
    head = head.next;
    return head;
}
```

TC = O(1)

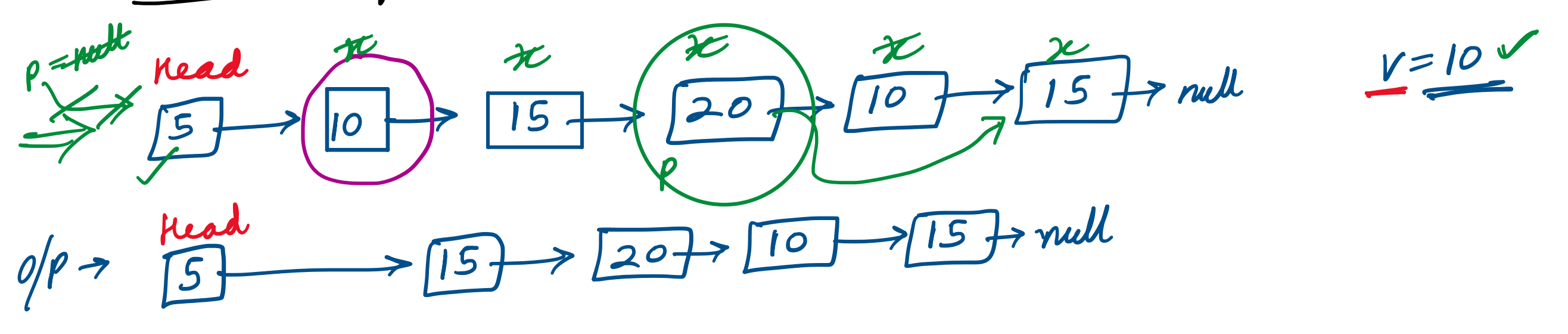


b) Tail node

```
if(head == null || head.next == null) return null;
node x = head;
while(x.next.next != null) { x = x.next; }
x.next = null;
return head;
```

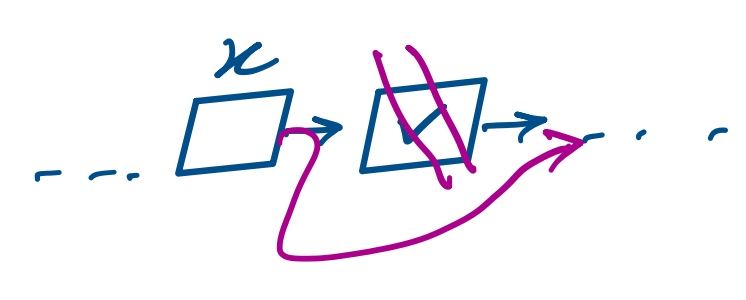
TC = O(N)

c) Delete the first occurrence of node with value V from LL.

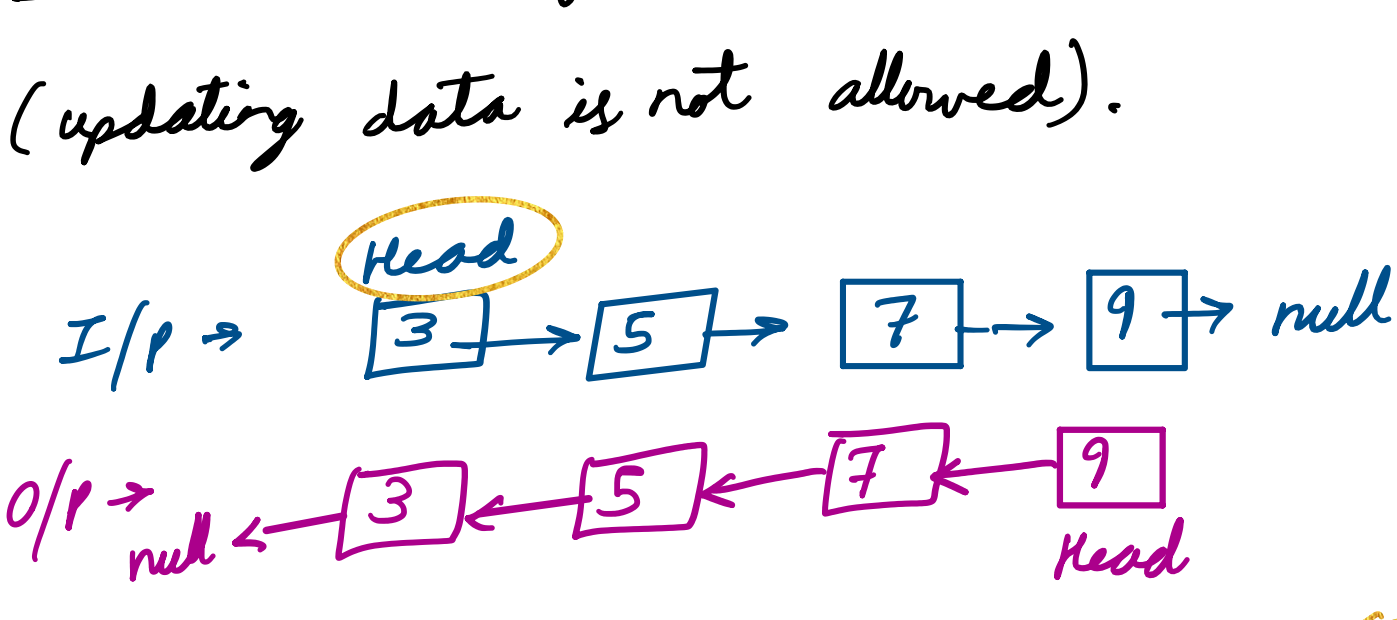


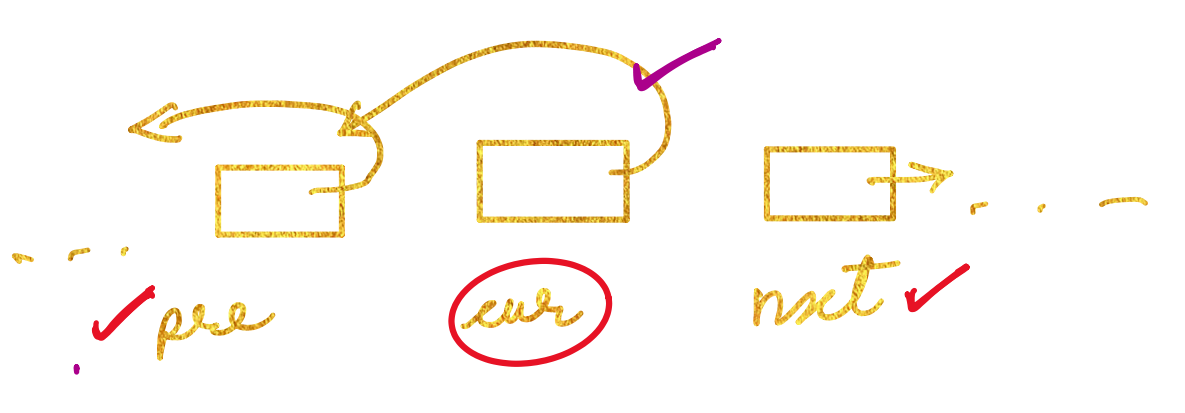

```
if(head == null) { return null; }
if(head.data == v) { head = head.next; return head; }
node x = head;
while(x.next != null) {
    if(x.next.data == v) {
        x.next = x.next.next;
        return head;
    }
    x = x.next;
}
return head;
```

TC = O(N)



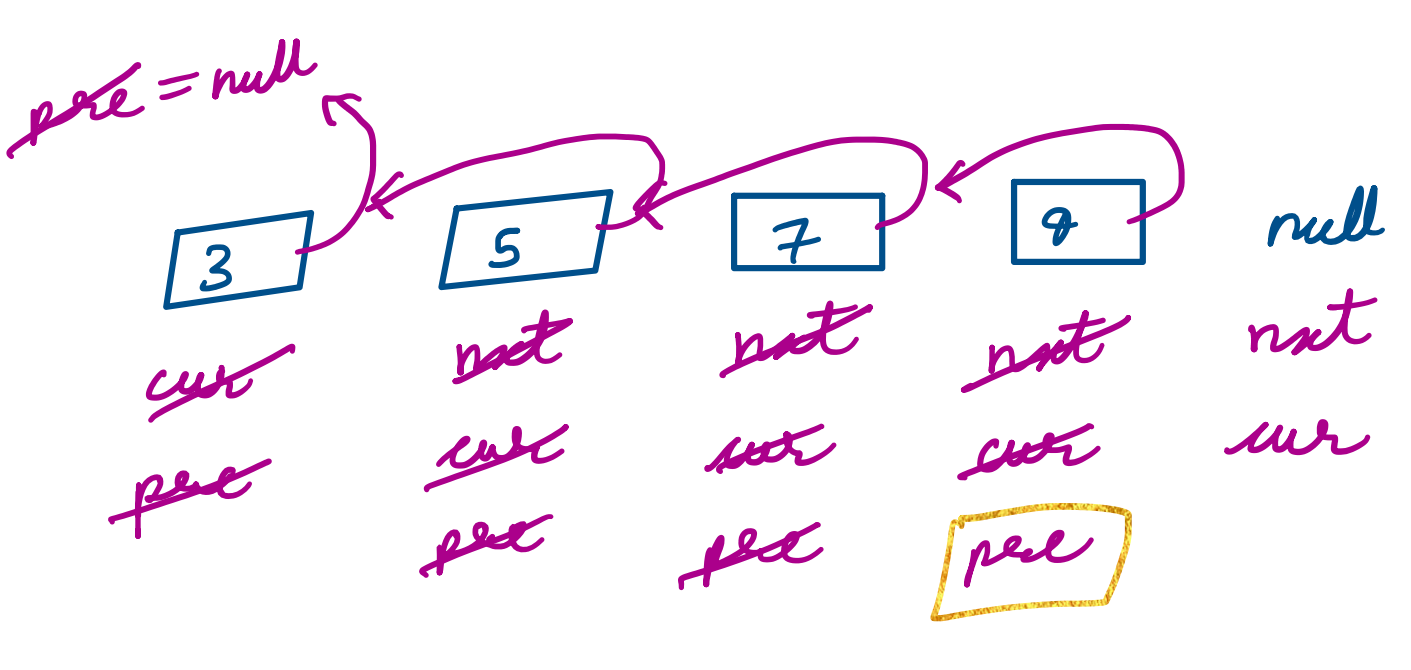
Q→ Reverse the given linked list in constant space.
(updating data is not allowed).





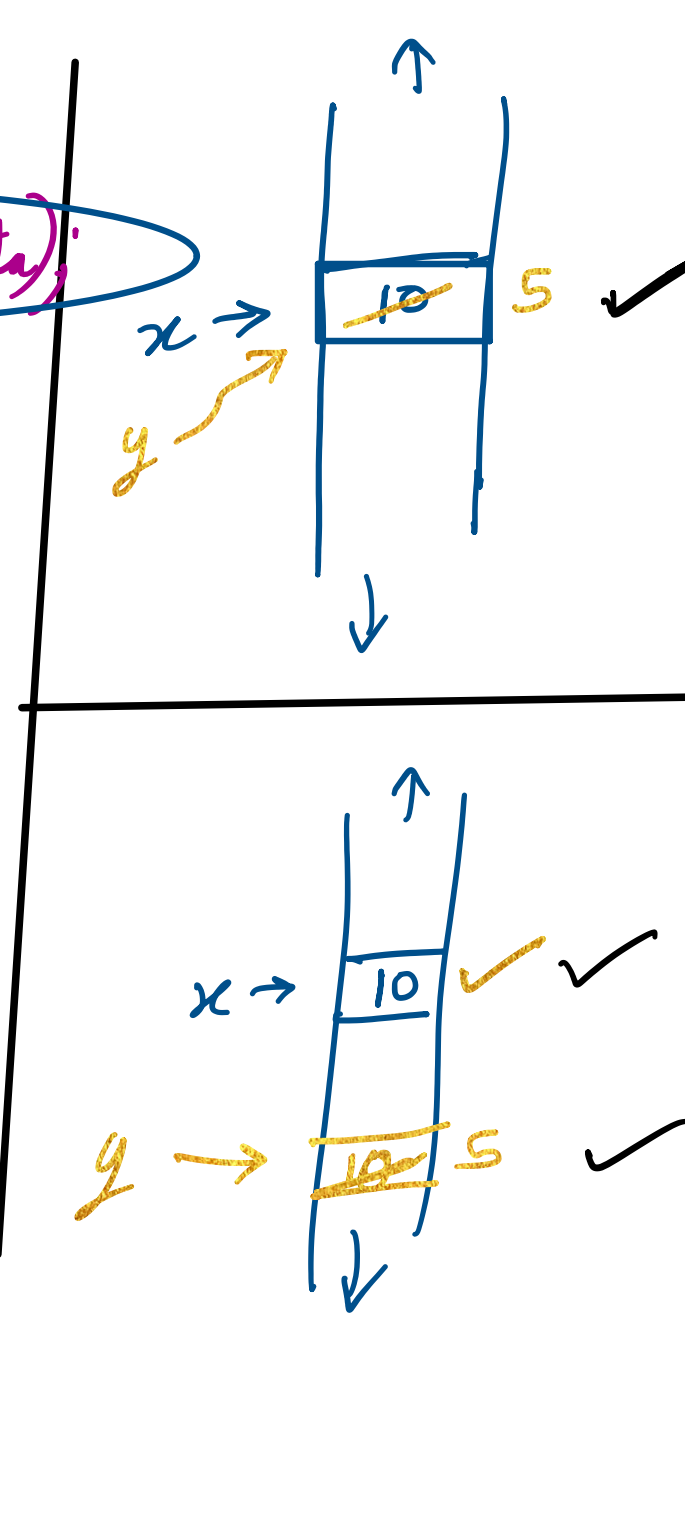

```
node pre = nxt = null;
node cur = head;
while(cur != null) {
    nxt = cur.next;
    cur.next = pre;
    pre = cur;
    cur = nxt;
}
return pre;
```

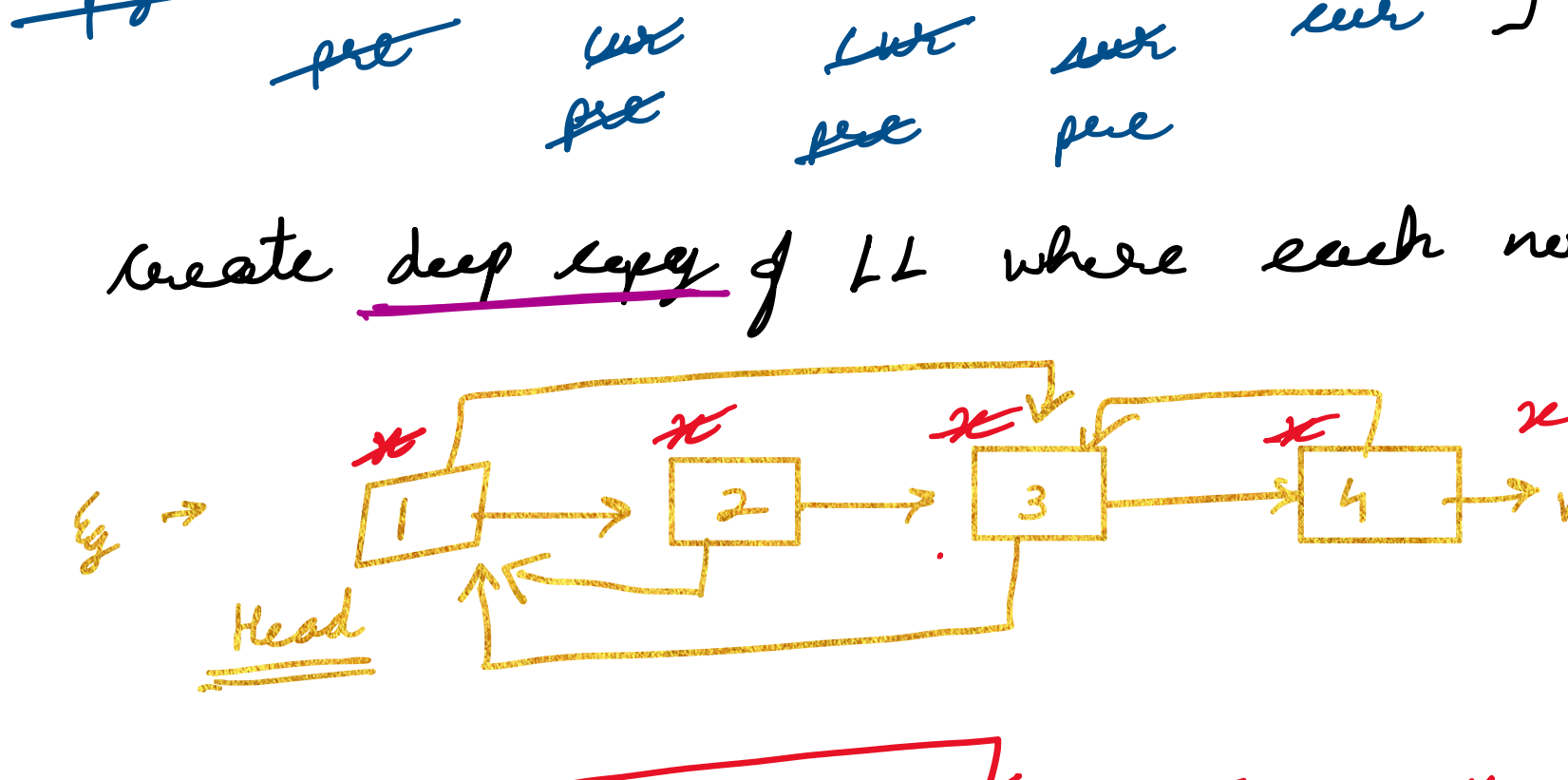
TC = O(N)



Q→

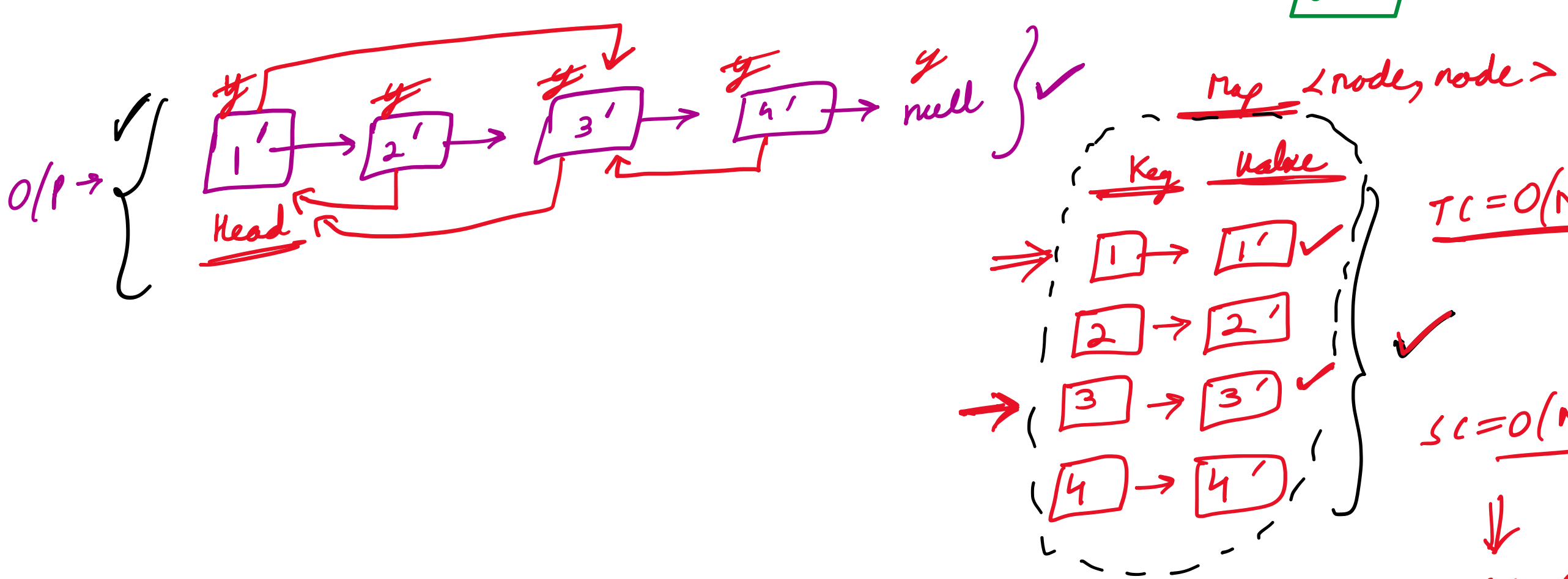
Shallow Copy	Deep Copy
node y = x;	node y = new node(x.data);
y.data = 5;	y.data = 5;
print(x.data)	print(x.data);
o/p → 5	o/p → 10



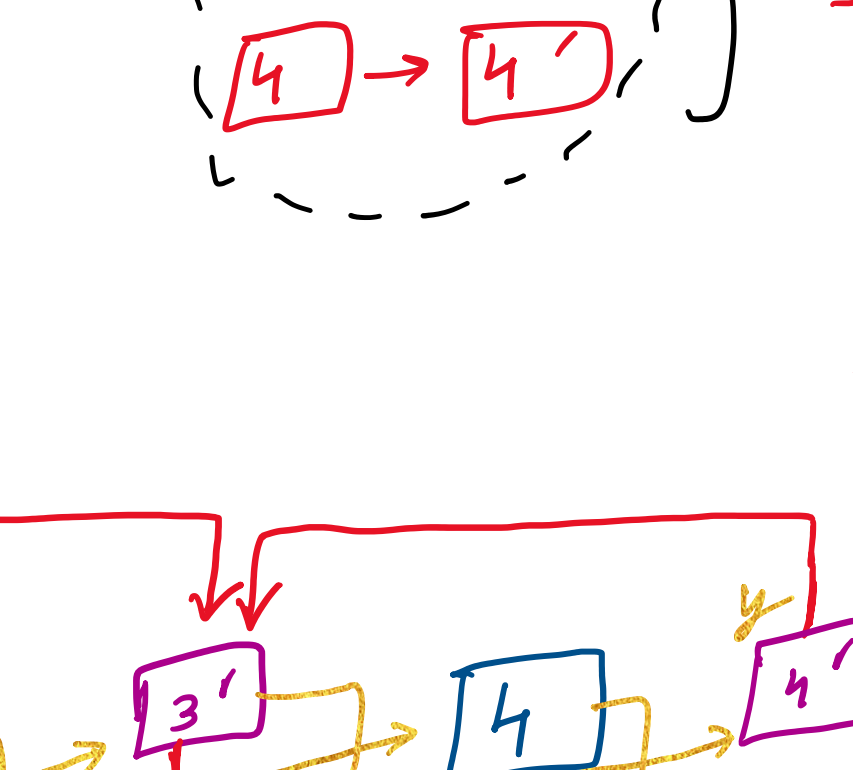


Create deep copy of LL where each node has next & random pointer.

25 Interview
Amazon *2

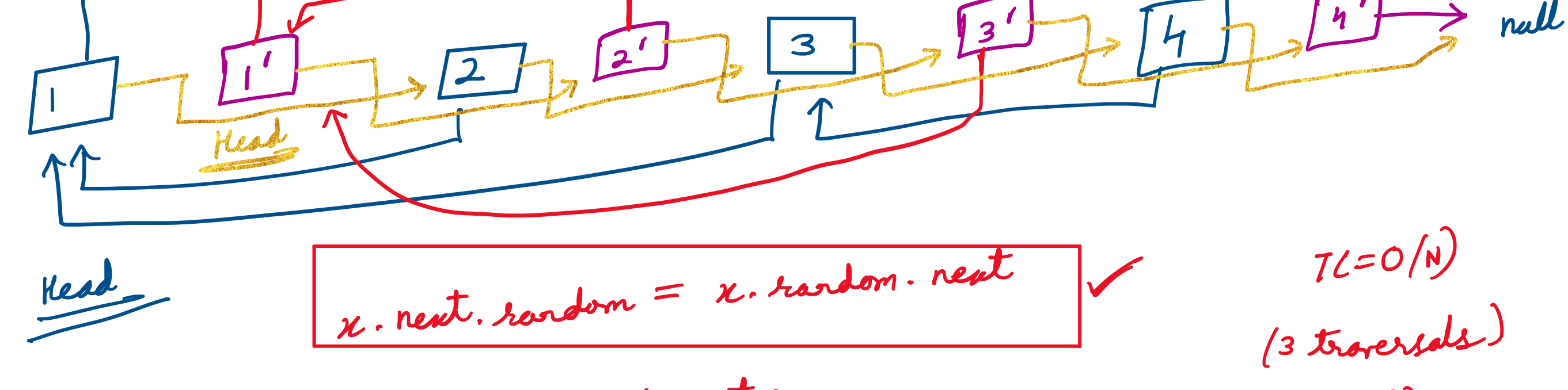


Map <node, node>



TC = O(N) ✓

SC = O(N) ✓
↓
O(1) ?



x.next.random = x.random.next

x = x.next.next;

TC = O(N)
(3 traversals)
SC = O(1)

x.next = x.next.next; x = x.next; if(x == null) return;
y.next = y.next.next; y = y.next;