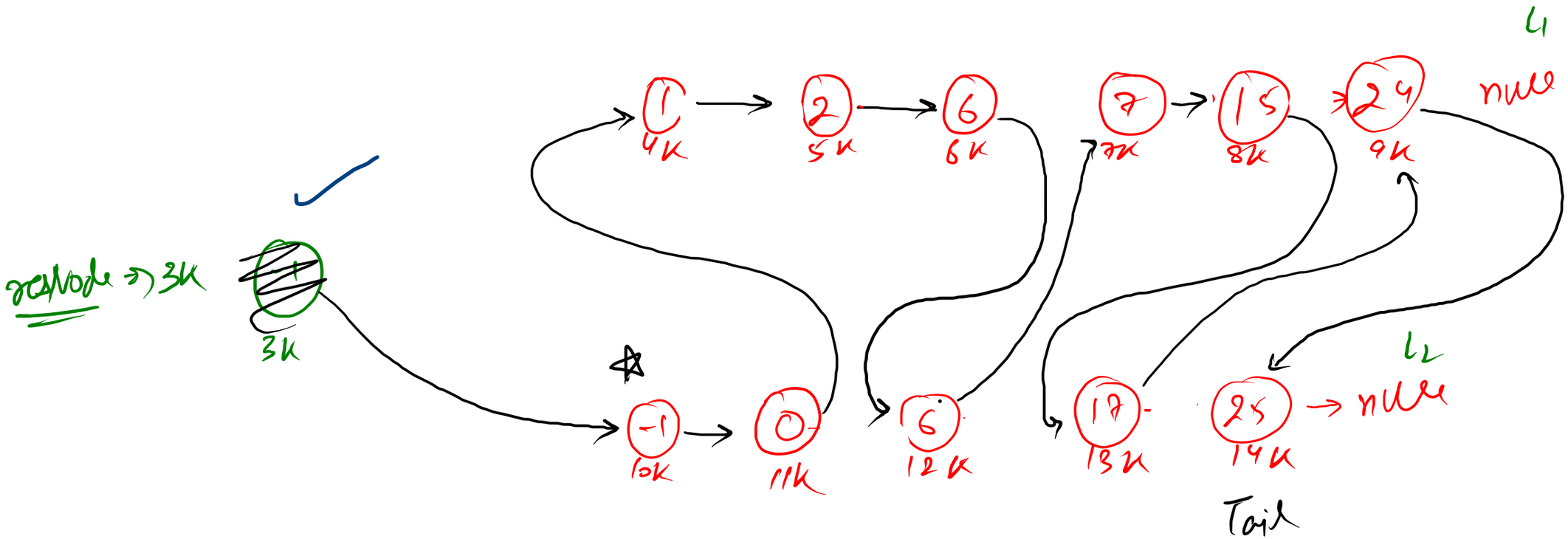
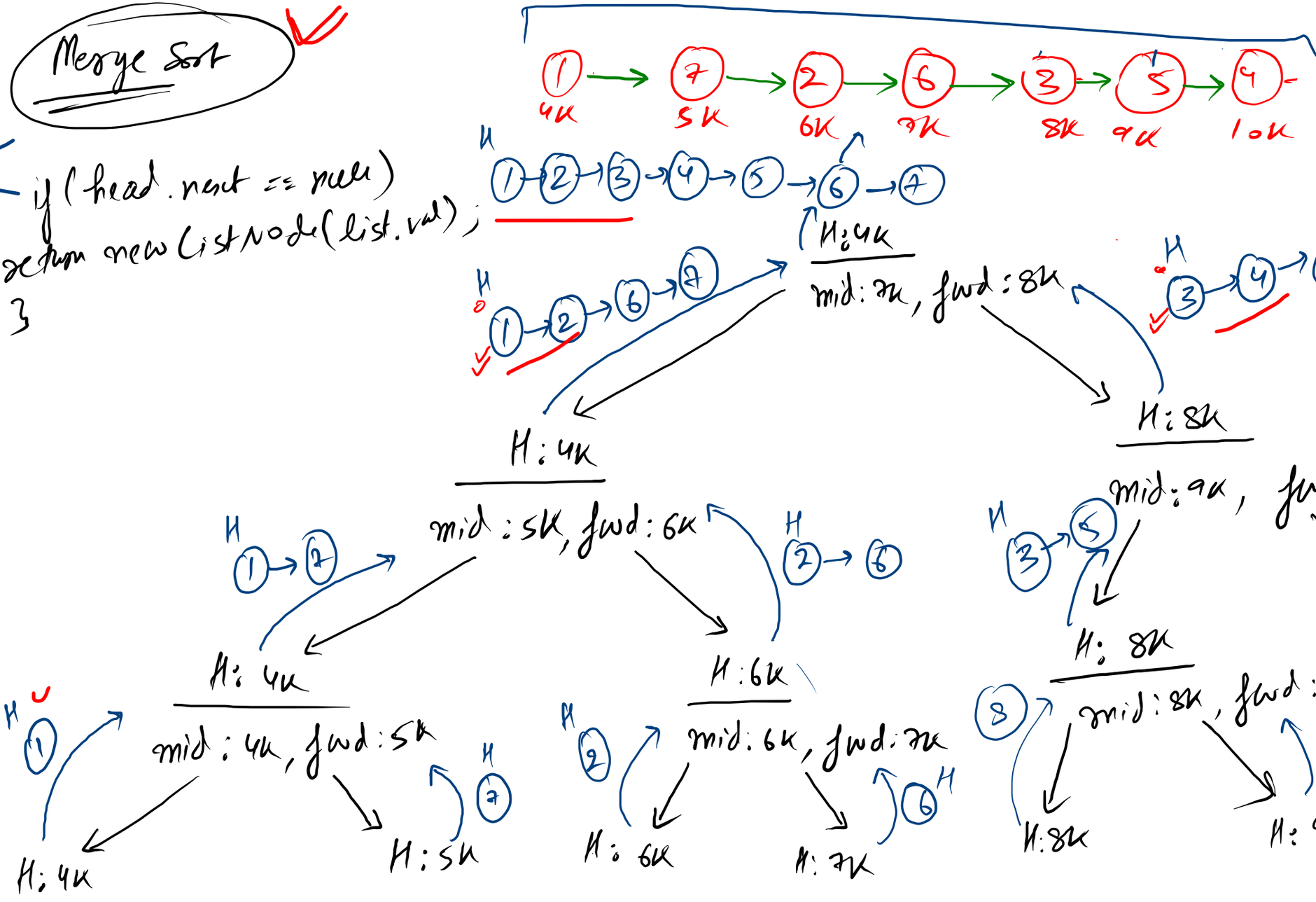

L1
1
4K
2
5K
6
6K
7
7K
15
8K
24
9K
null
getNode() → 3K
1
3K
-1
10K
0
11K
6
12K
17
13K
25
14K
null
L2
Tail

Merge Sort

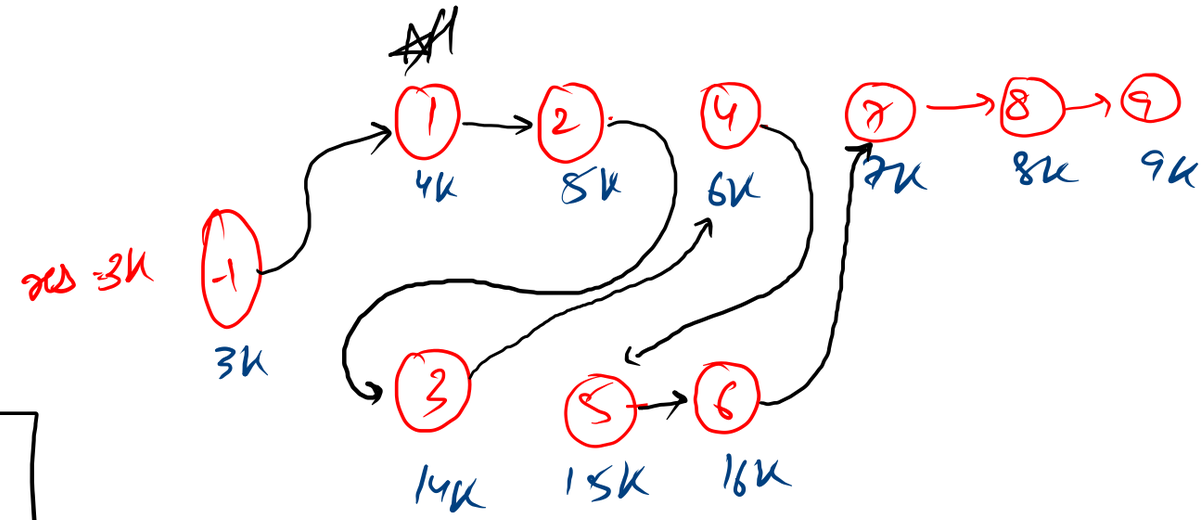
```
if (head.next == null)
return new ListNode(list.val);
}
```



```java
public static ListNode mergeTwoLists(ListNode l1, ListNode l2) {
    if(l1 == null && l2 == null){
        return null;
    }

    ListNode res = new ListNode(-1); //  dummy
    ListNode tail = res;

    while(l1 != null && l2 != null){
        if(l1.val < l2.val){
            tail.next = l1;
            l1 = l1.next;
        }else{
            tail.next = l2;
            l2 = l2.next;
        }
        tail = tail.next;
    }

    if(l1 != null){
        tail.next = l1;
    }else{
        tail.next = l2;
    }

    return res.next;
}
```

T.C. ⇒ O(n+m)

S.C. ⇒ O(1)



```java
public static ListNode mergeSort(ListNode head) {
    if(head == null){
        return null;
    }

    if(head.next == null){
        return new ListNode(head.val);
    }

    ListNode midNode = midNodeOfLinkedList(head);
    ListNode fwd = midNode.next;
    midNode.next = null;

    ListNode lHead = mergeSort(head);
    ListNode rHead = mergeSort(fwd);

    midNode.next = fwd;

    return mergeTwoLists(lHead,rHead);
}
```

T(n)

O(1)

O(1)

O(n)

O(1)

T(n/2)

T(n/2)

O(1)

O(n)

$T(n) \Rightarrow 2T(n/2) + 2 \cdot n$

T.C. $\Rightarrow$?

O(n)

n/60000

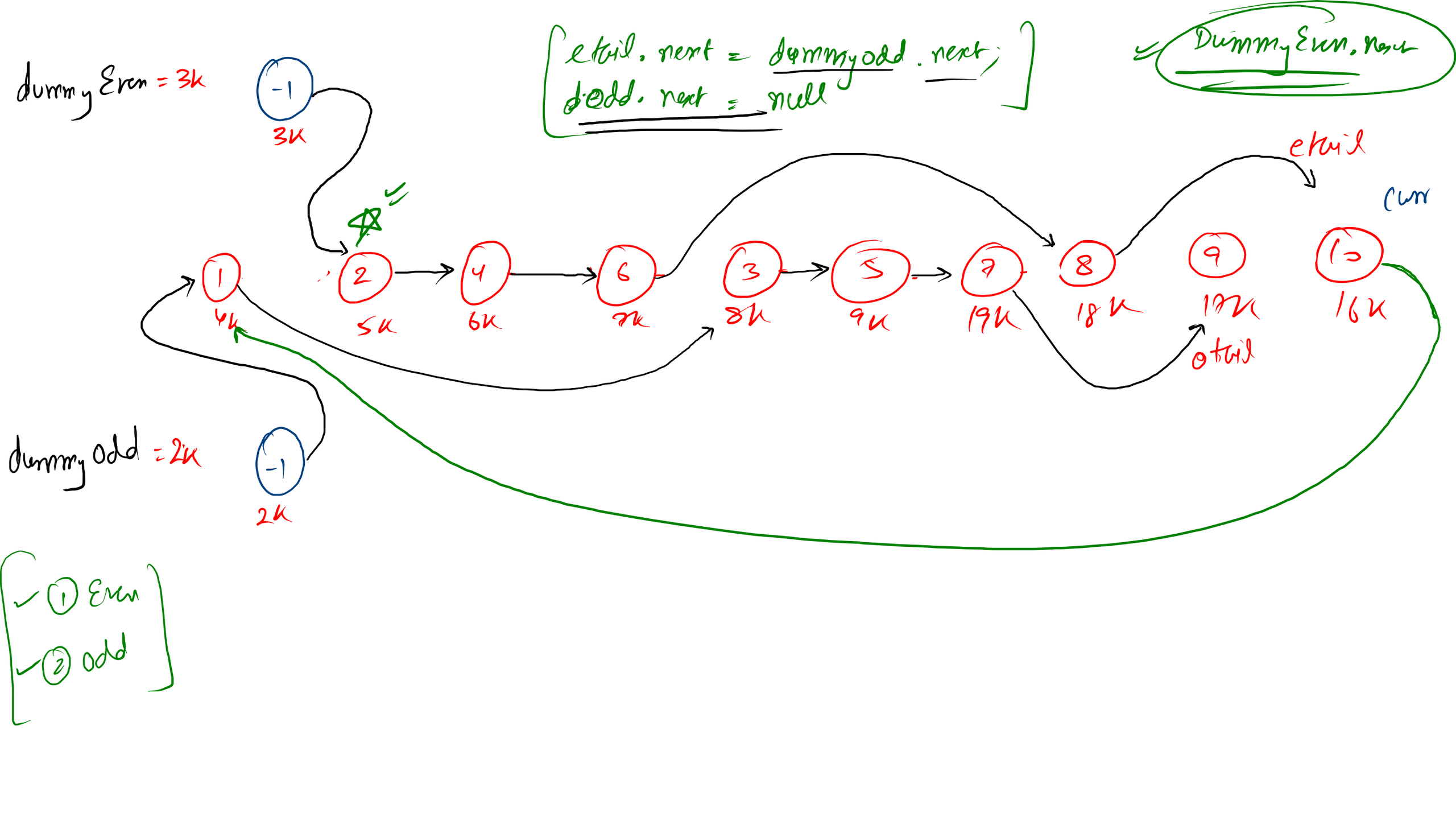
n

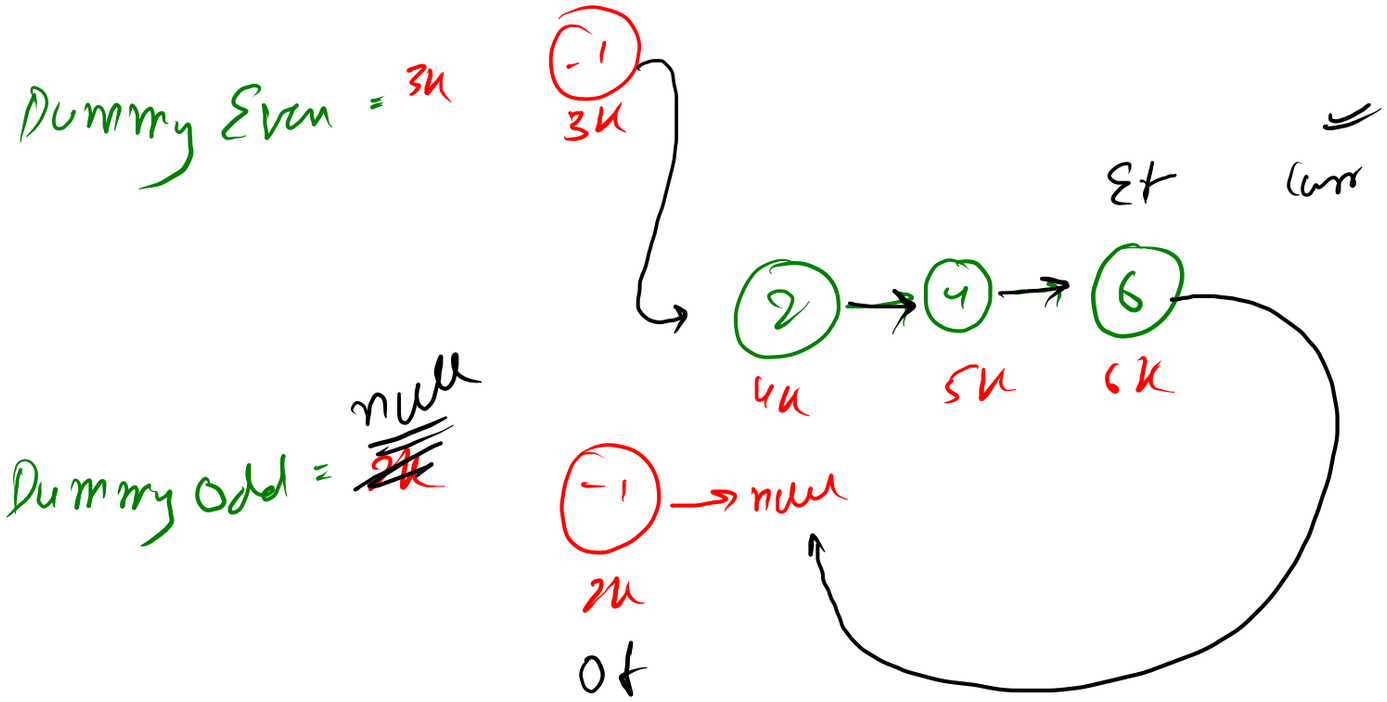
n

2

n/4

n/1000

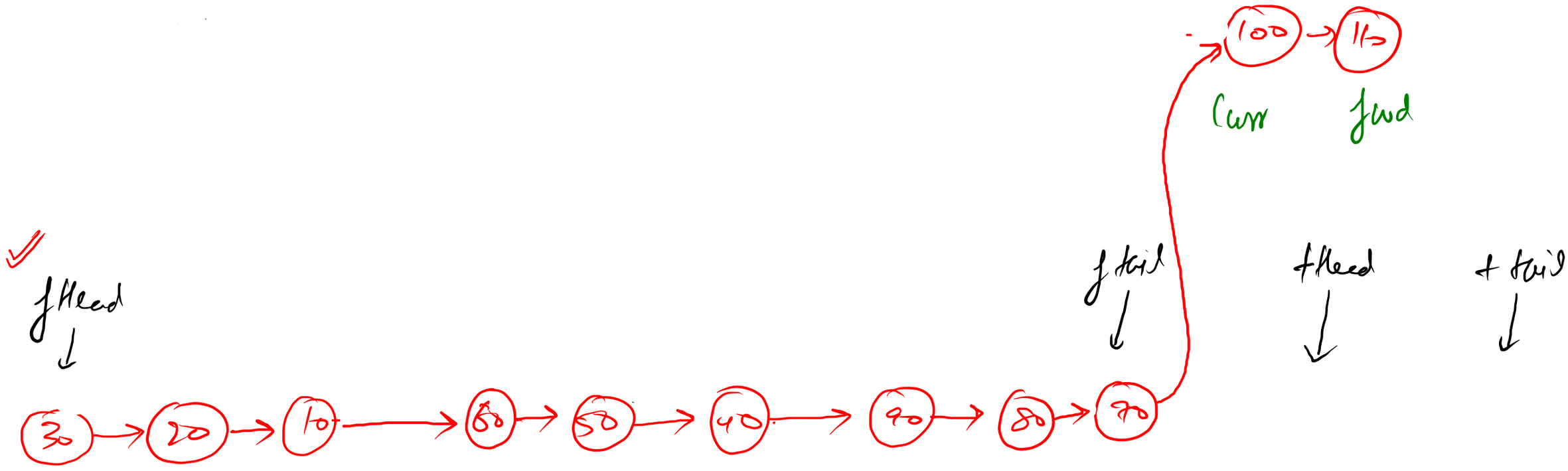

dummyEven = 3k
-1
3k
etail.next = dummyodd.next;
dodd.next = null
DummyEven.next
etail
curr
1
4k
2
5k
4
6k
6
7k
3
8k
5
9k
7
19k
8
18k
9
17k
10
16k
otail
dummyodd = 2k
-1
2k
1 Even
2 odd


Dummy Even = 3k
-1
3k
2
4k
4
5k
6
6k
Et
curr
Dummy Odd = null
-1
null
2k
Ot

Size = 11 - 3 - 3 - 3

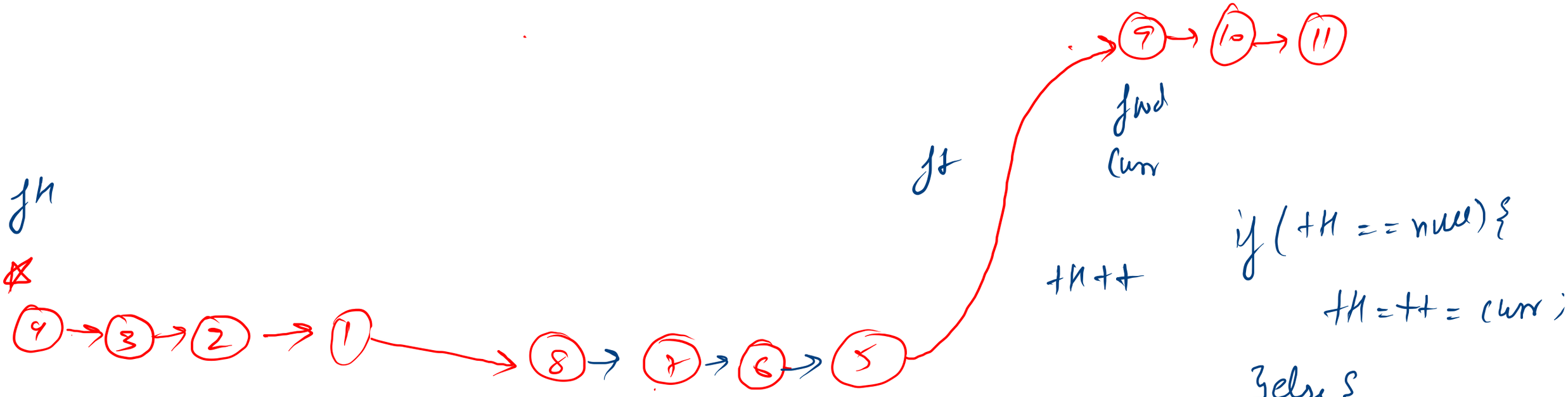
=> 2

K = 3



Size = 13

K = 4



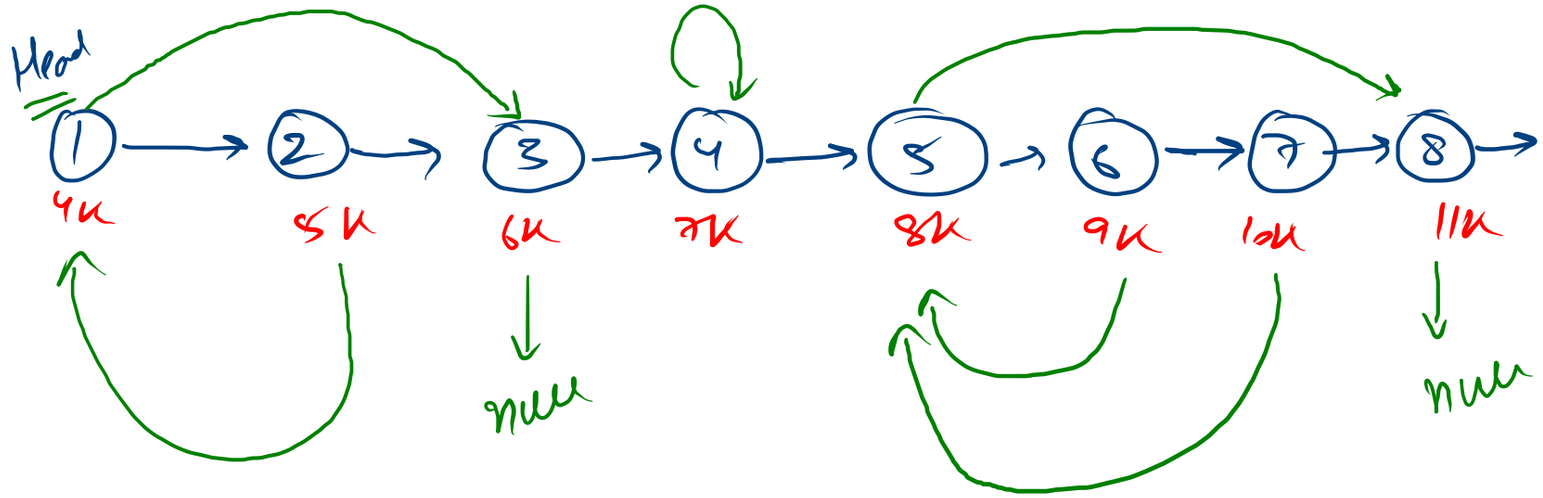

```
if (fH == null) {
    fH = tH;
    ft = ft;
    tH = tt = null;
}
```

Else

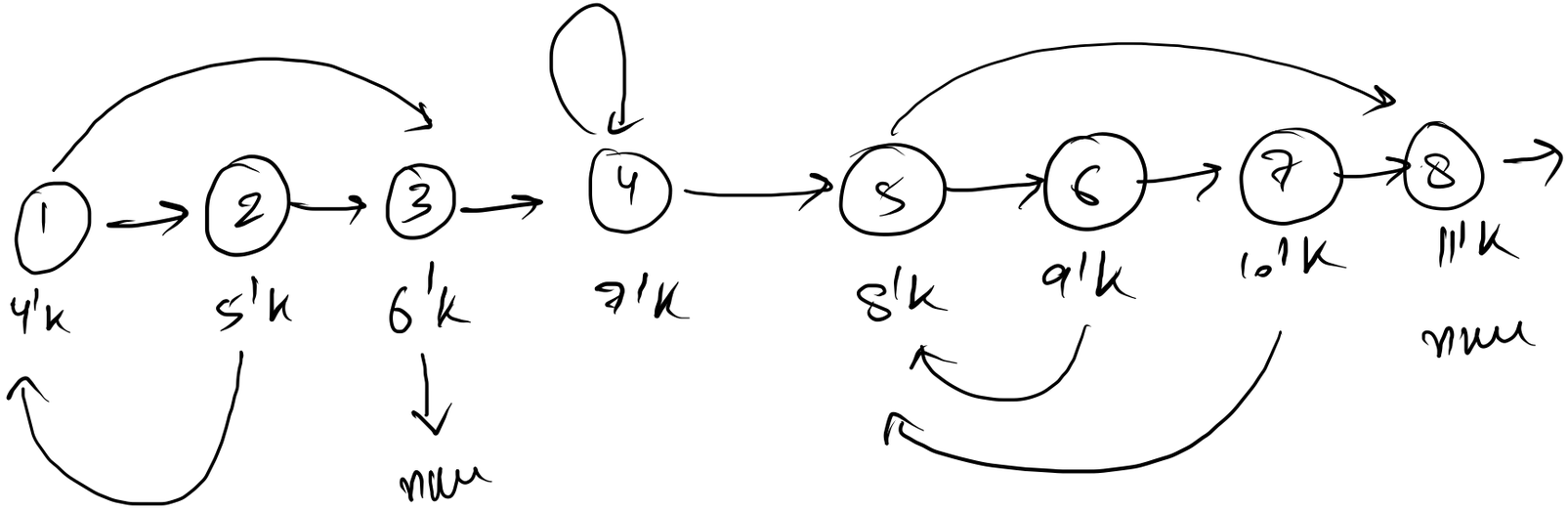
```
{
    ft.next = tH;
    ft = tt;
}
```

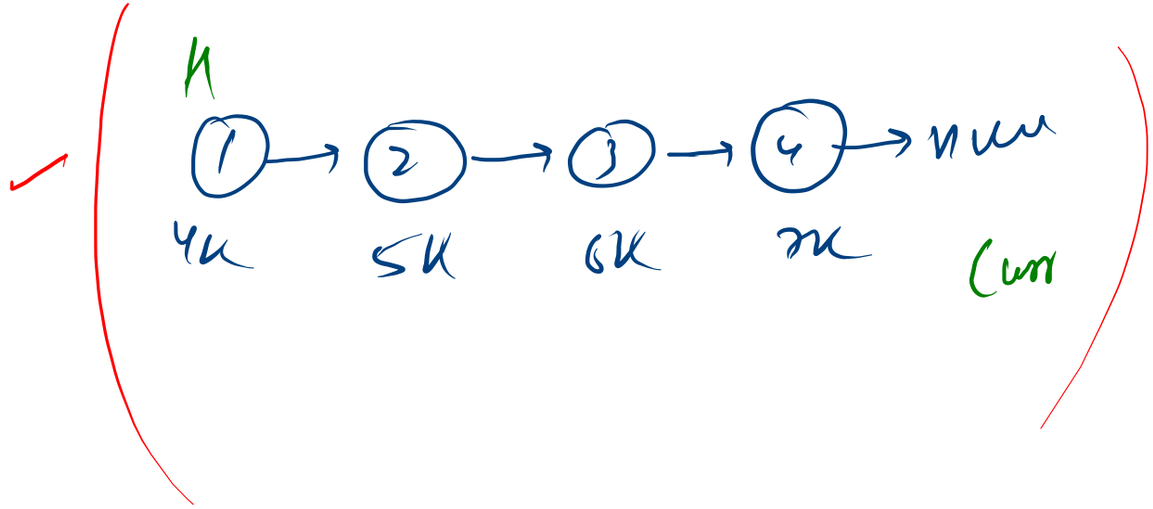
```
if (tH == null) {
    tH = tt = curr;
} else {
    curr.next = tH;
    tH = curr;
}
```

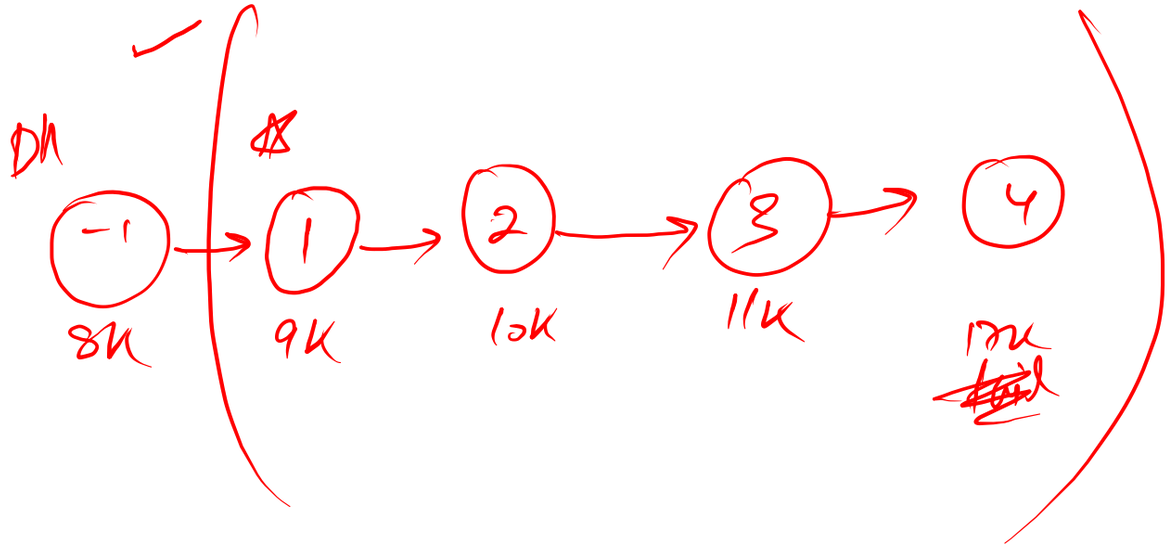
Input

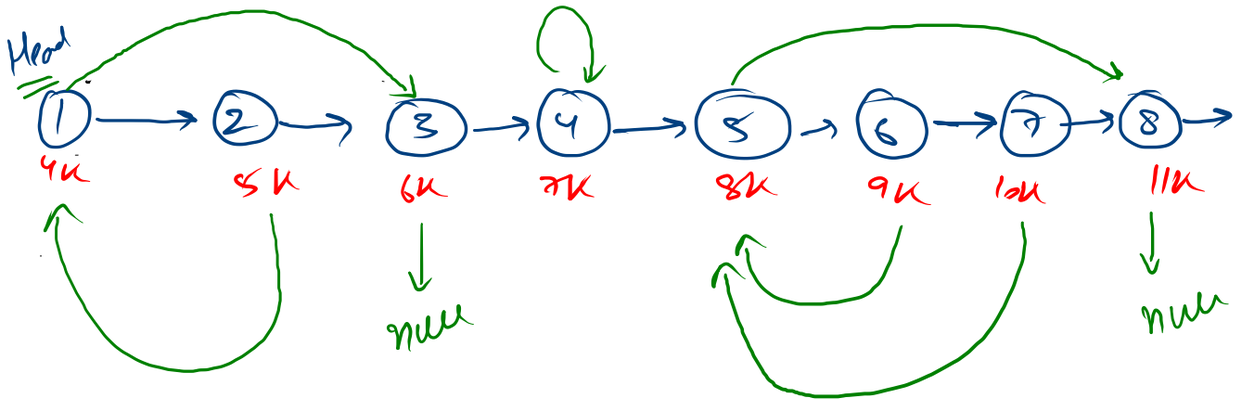


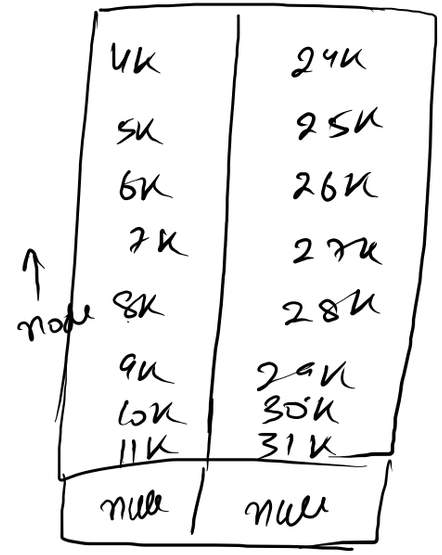
Output



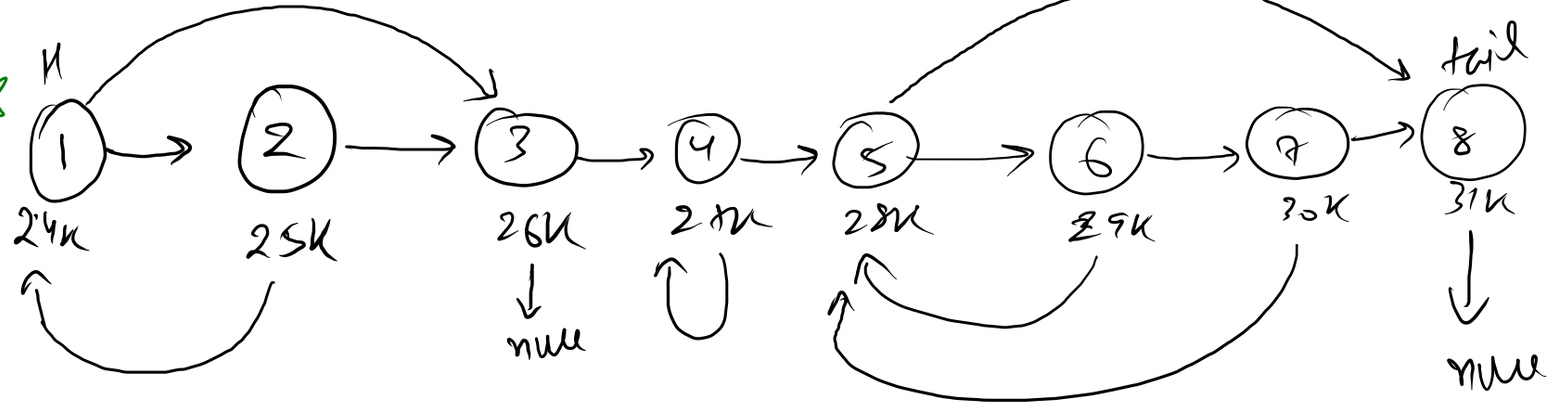

1
2
3
4
4K
5K
6K
7K


-1
1
2
3
4
8K
9K
10K
11K
12K

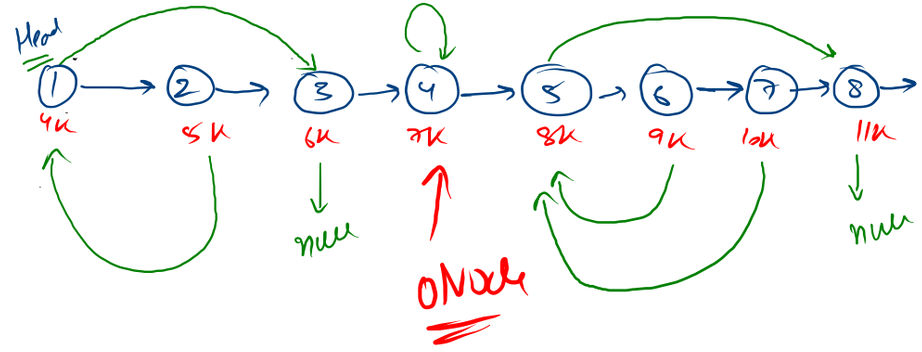


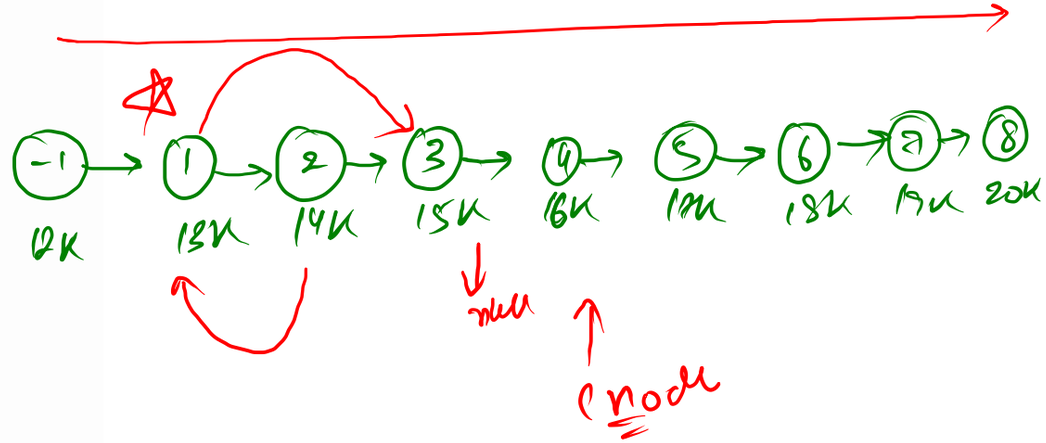



```
(node.random = map.get(node.random)
```



```java
public static ListNode copyRandomList(ListNode head) {
    if(head == null){
        return null;
    }

    HashMap<ListNode,ListNode> map = new HashMap<>();
    map.put(null,null);
    ListNode dummy = new ListNode(-1);
    ListNode curr = head , prev = dummy;

    while(curr != null){
        ListNode node = new ListNode(curr.val);
        prev.next = node;
        map.put(curr,node);
        prev = prev.next;
        curr = curr.next;
    }

    ListNode onode = head , cnode = dummy.next;

    while(onode != null){
        cnode.random = map.get(onode.random);

        cnode = cnode.next;
        onode = onode.next;
    }

    return dummy.next;
}
```



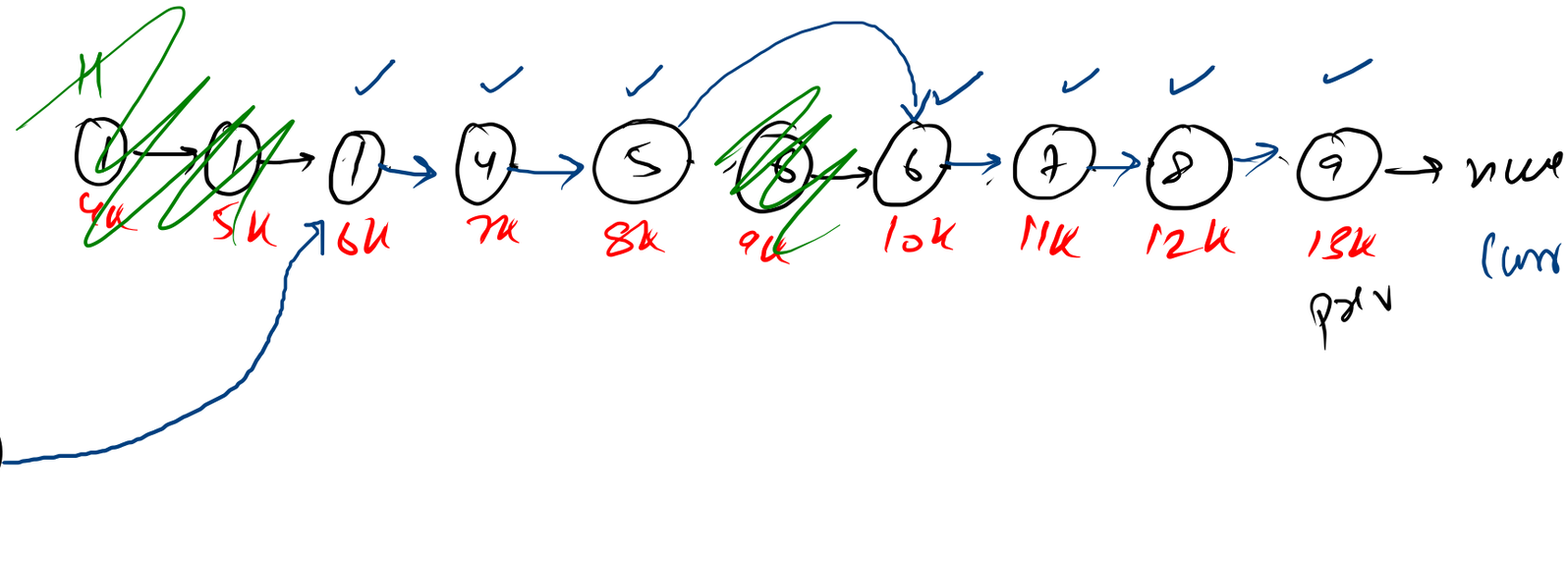


dummy = 12 K

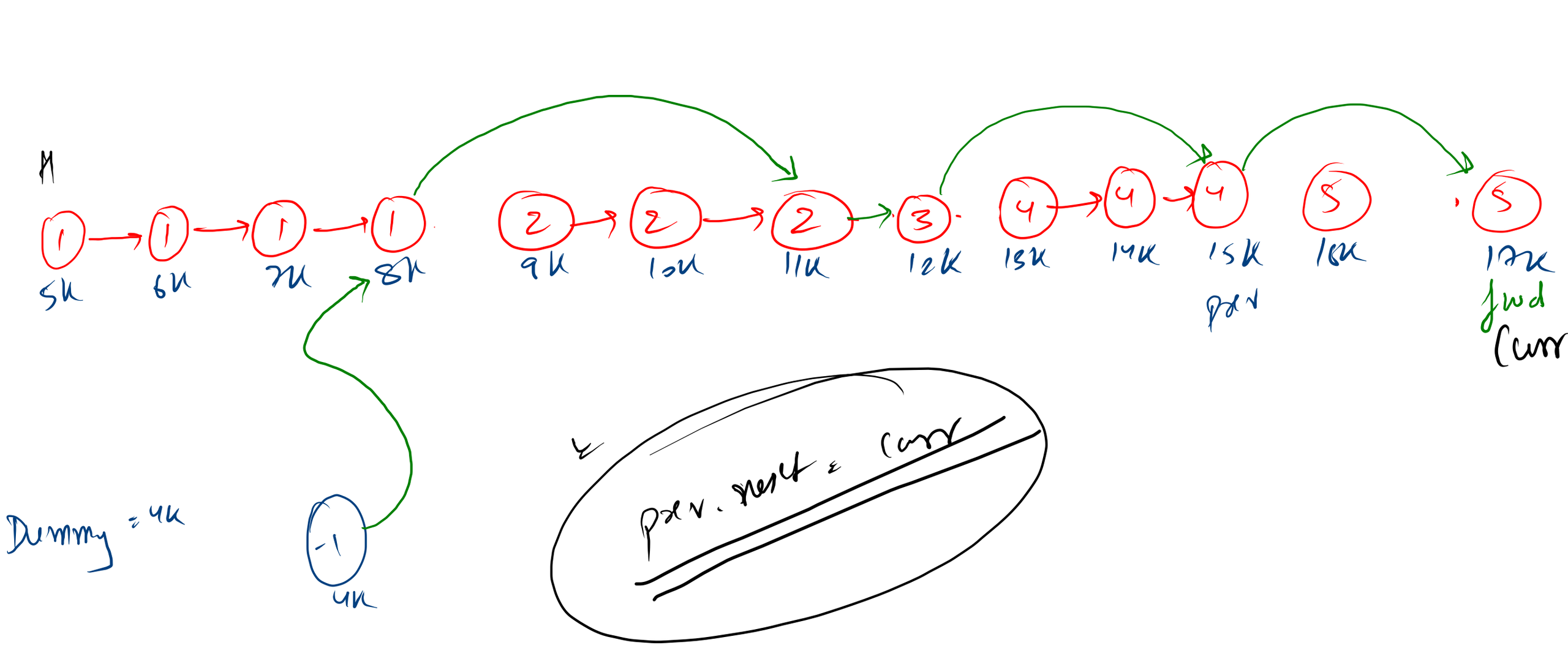
null	→	null
4K	→	13K
5K	→	14K
6K	→	15K
7K	→	16K
8K	→	17K
9K	→	18K
10K	→	19K
11K	→	20K

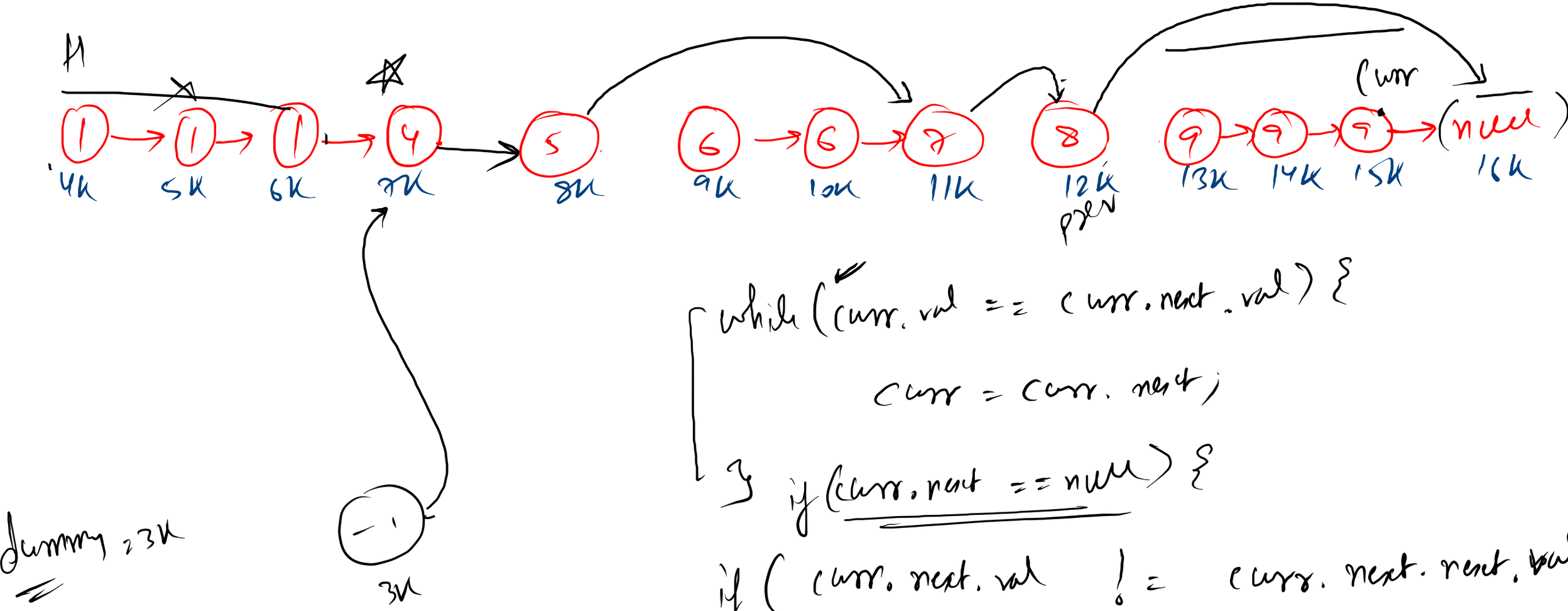
Sorted

LL

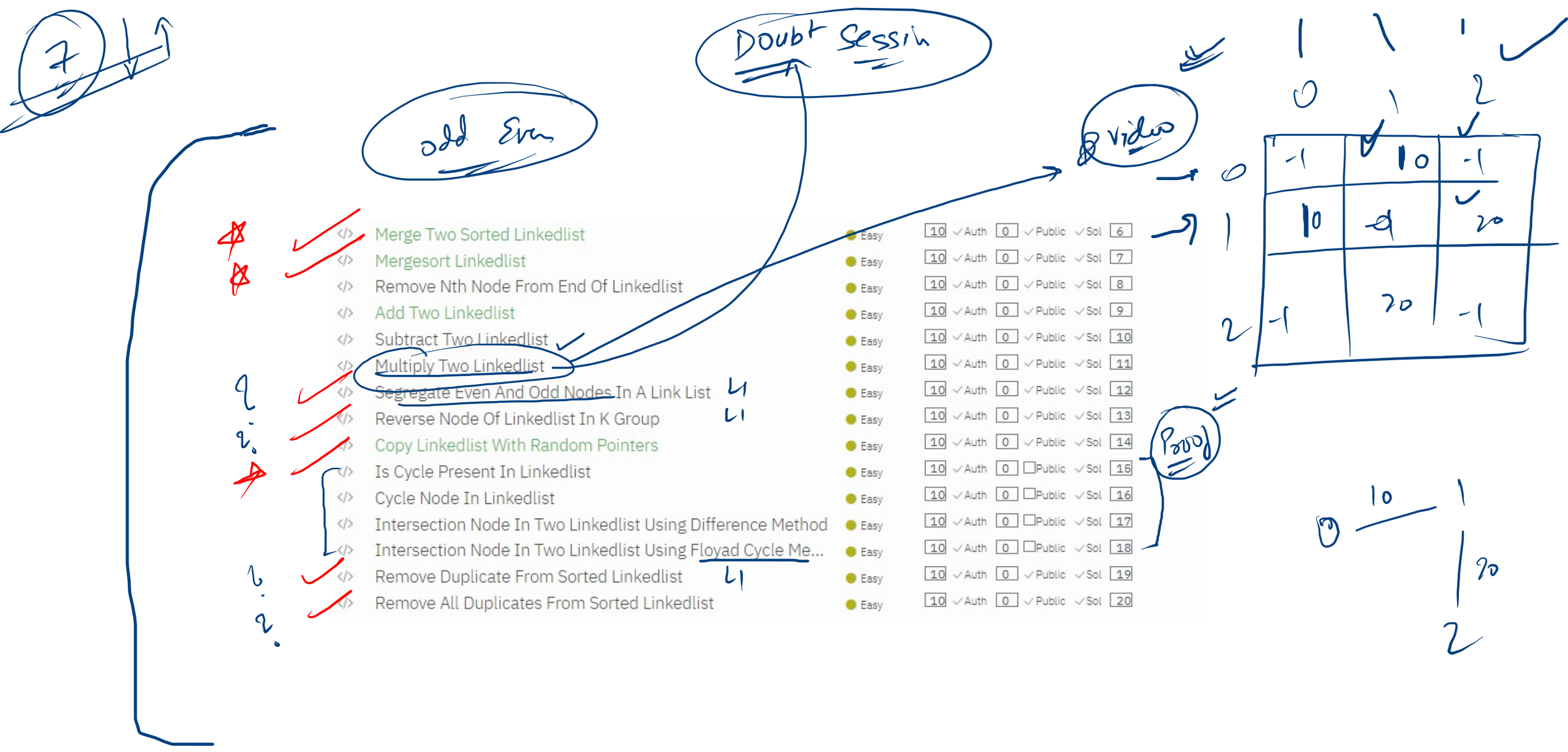


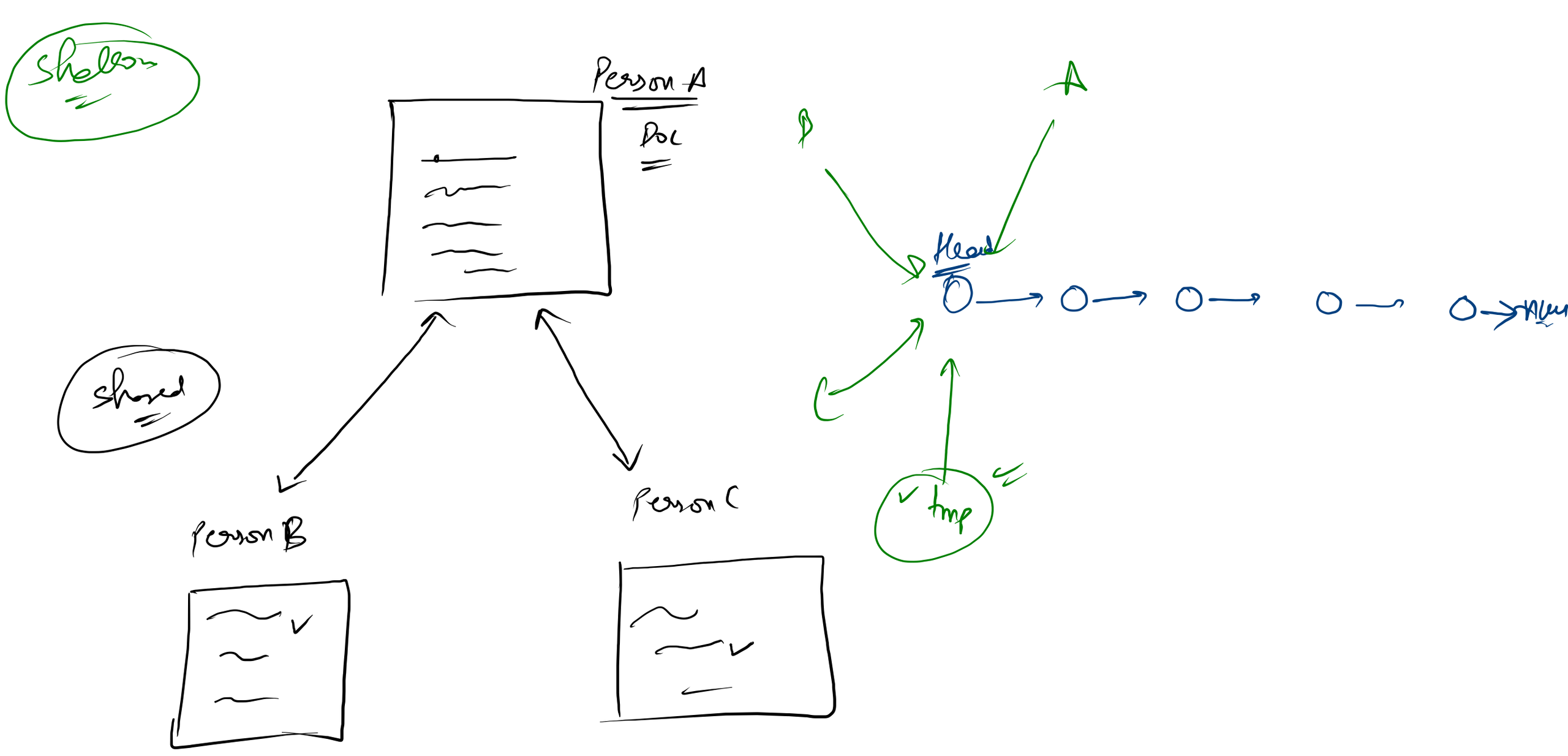
Dummy -> 4K

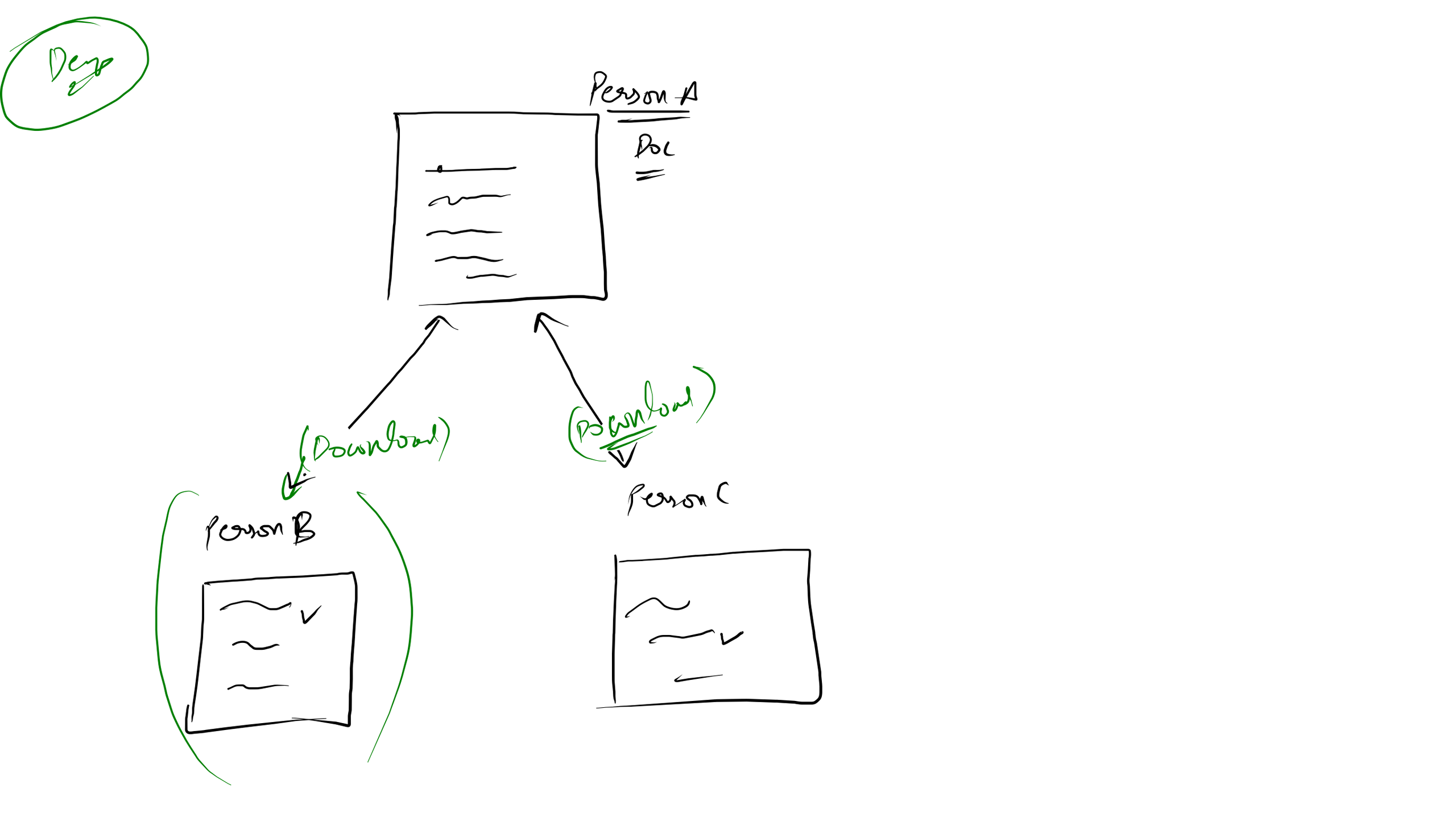

1
1
1
1
2
2
2
3
4
4
4
5
5
5K
6K
7K
8K
9K
10K
11K
12K
13K
14K
15K
16K
17K
prev
fwd
curr
Dummy = 4K
-1
4K
prev.next = curr




```
while (curr.val == curr.next.val) {
    curr = curr.next;
} if (curr.next == null) {
if ( curr.next.val != curr.next.next.val){
    prev.next = curr.next;
    prev = prev.next;
}
```


Doubt Session
odd Even
video
Poool
Merge Two Sorted Linkedlist
Mergesort Linkedlist
Remove Nth Node From End Of Linkedlist
Add Two Linkedlist
Subtract Two Linkedlist
Multiply Two Linkedlist
Segregate Even And Odd Nodes In A Link List
Reverse Node Of Linkedlist In K Group
Copy Linkedlist With Random Pointers
Is Cycle Present In Linkedlist
Cycle Node In Linkedlist
Intersection Node In Two Linkedlist Using Difference Method
Intersection Node In Two Linkedlist Using Floyad Cycle Me...
Remove Duplicate From Sorted Linkedlist
Remove All Duplicates From Sorted Linkedlist
Easy
10 Auth 0 Public Sol 6
10 Auth 0 Public Sol 7
10 Auth 0 Public Sol 8
10 Auth 0 Public Sol 9
10 Auth 0 Public Sol 10
10 Auth 0 Public Sol 11
10 Auth 0 Public Sol 12
10 Auth 0 Public Sol 13
10 Auth 0 Public Sol 14
10 Auth 0 Public Sol 15
10 Auth 0 Public Sol 16
10 Auth 0 Public Sol 17
10 Auth 0 Public Sol 18
10 Auth 0 Public Sol 19
10 Auth 0 Public Sol 20


Sheldon
Person A
Doc
A
Head
shared
Person B
Person C
tmp


Person A
Doc
(Download)
(Download)
Person B
Person C