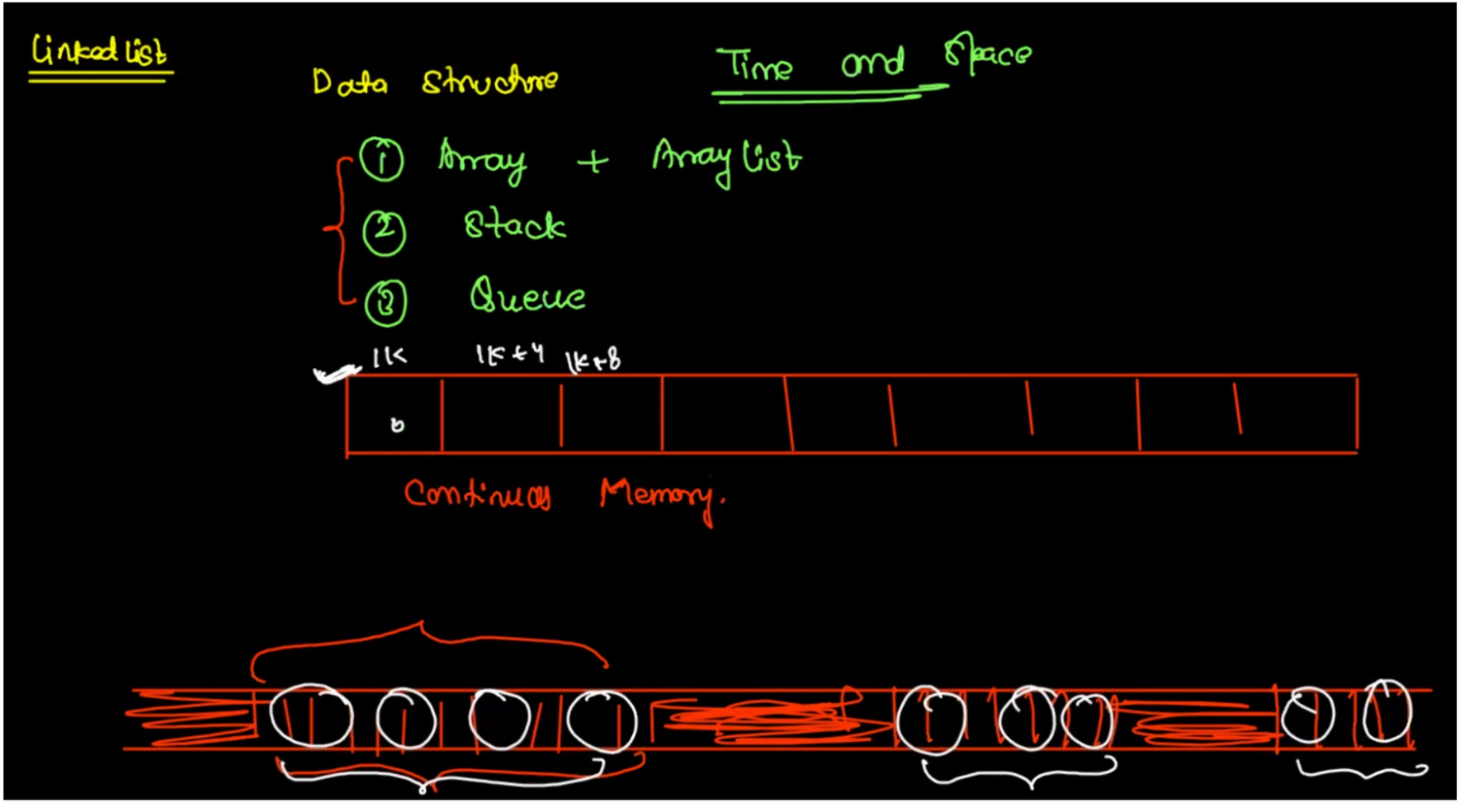
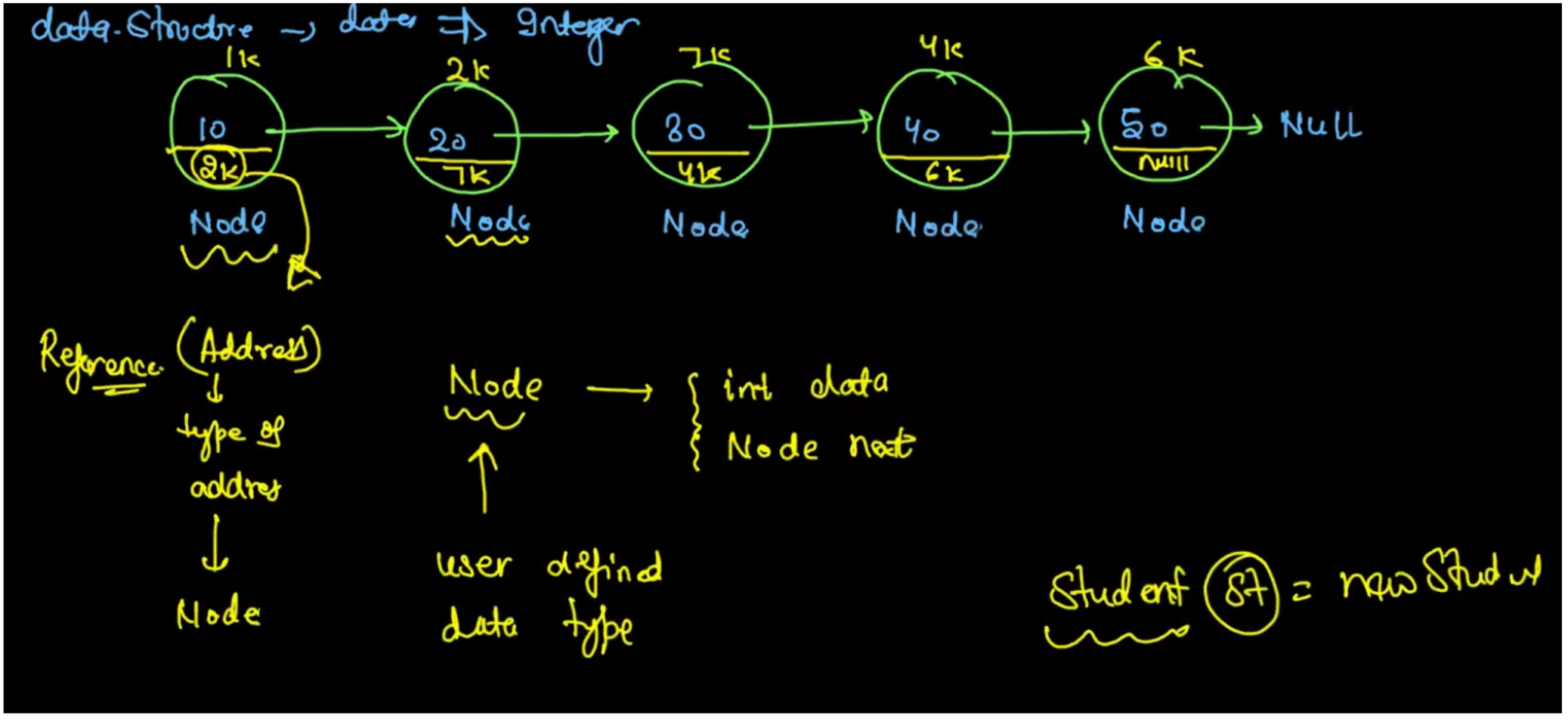
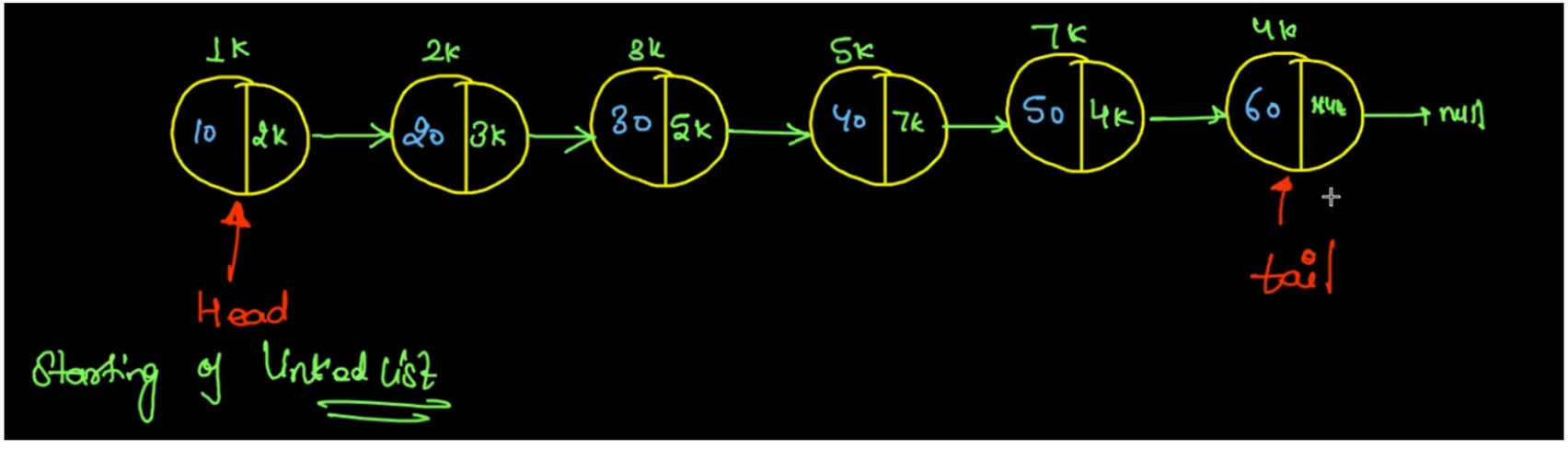
LINKED LIST



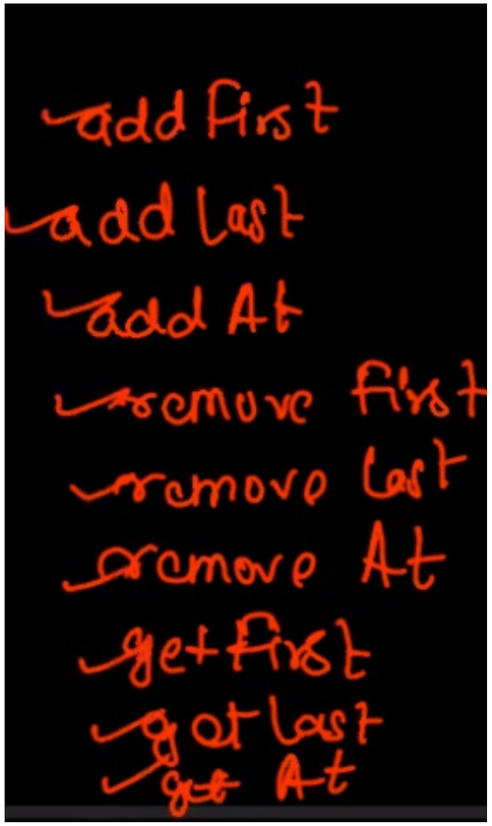
Make use of available small memory rather than continuous memory

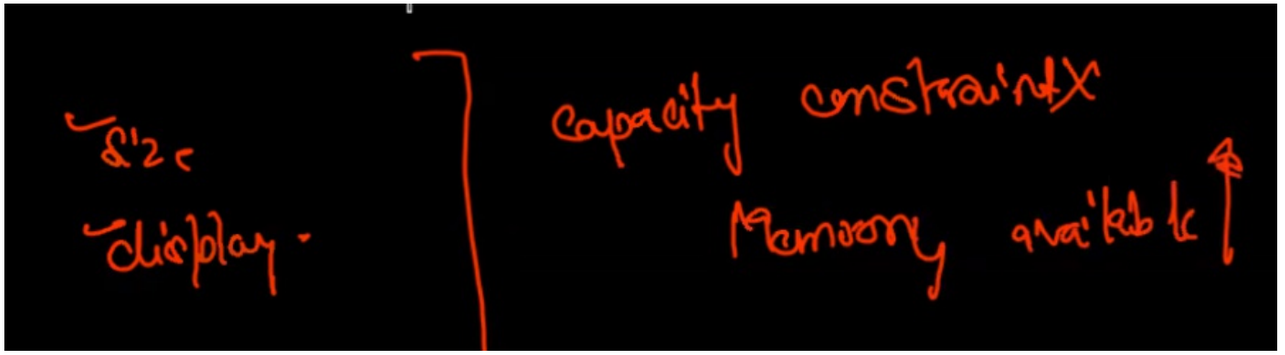

data Structure → data ⇒ Integer
1k
2k
7k
4k
6k
10
2k
20
7k
30
4k
40
6k
50
null
Null
Node
Node
Node
Node
Node
Reference (Address)
type of address
Node
Node → { int data
Node next
user defind data type
Student s7 = new Stud


1k
2k
3k
5k
7k
4k
10
2k
20
3k
30
5k
40
7k
50
4k
60
null
null
Head
tail
Starting of Linked list

size of node?

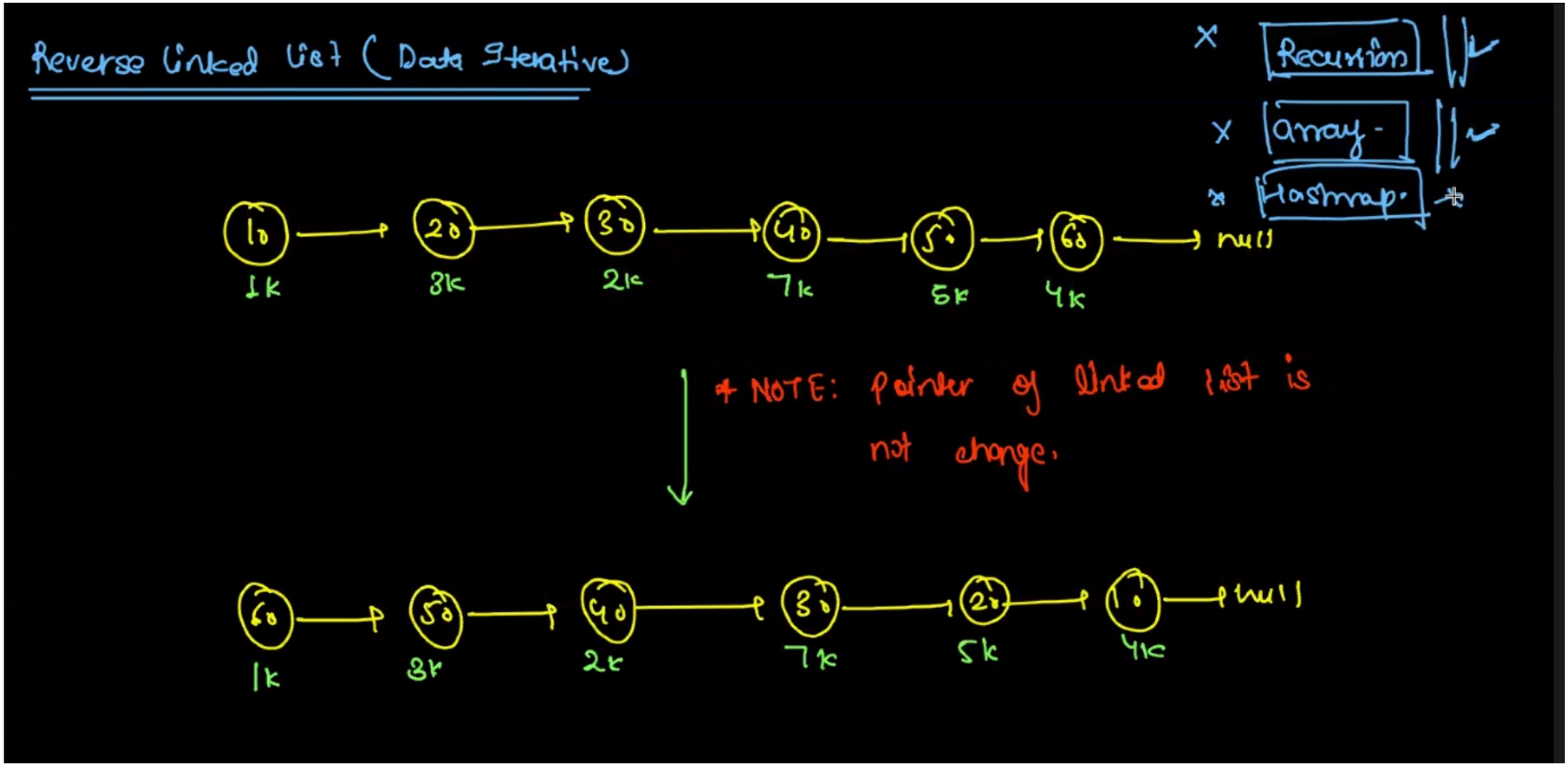
expected from user





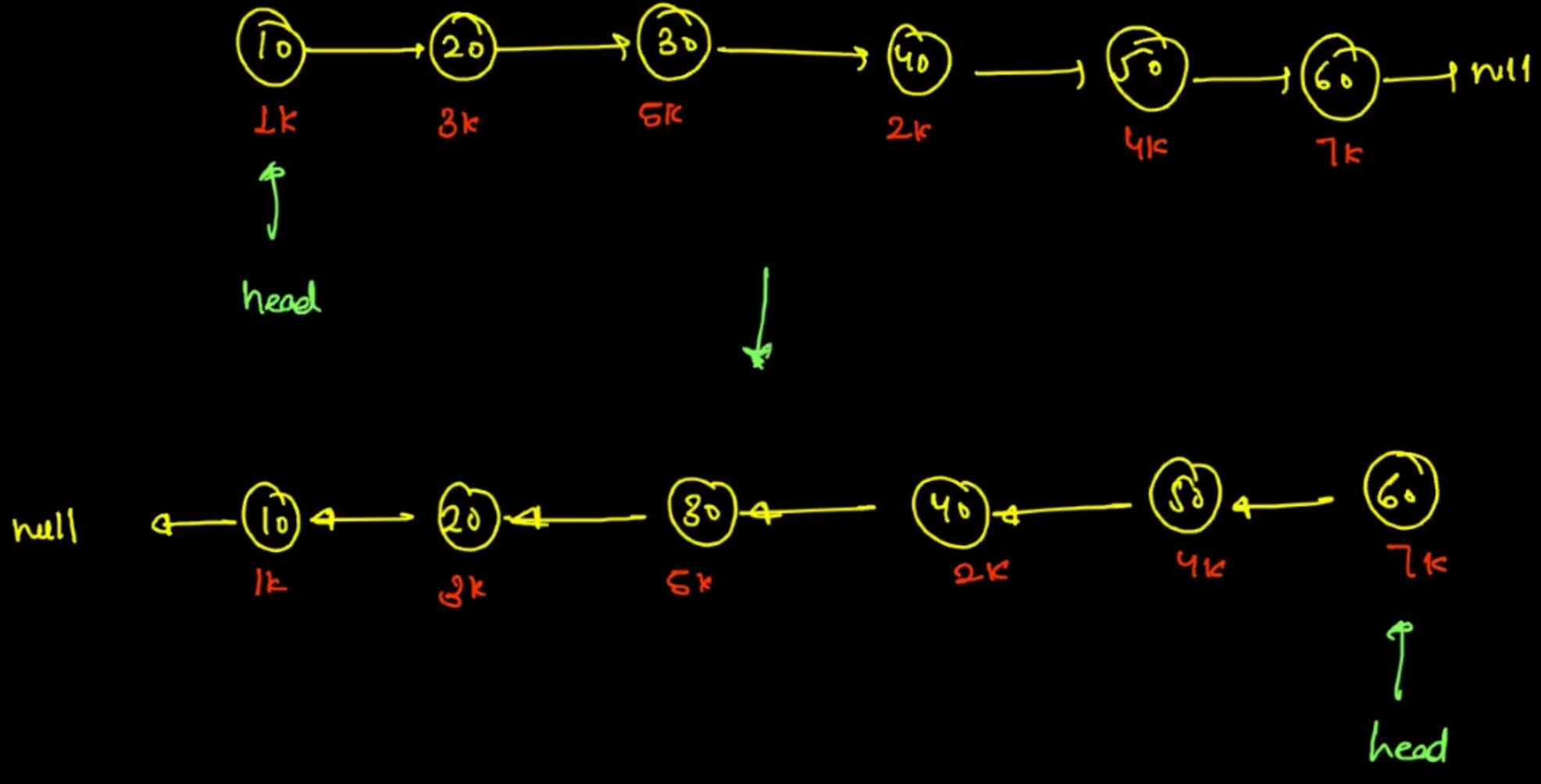
collection in java -> linklest is doubly linklist

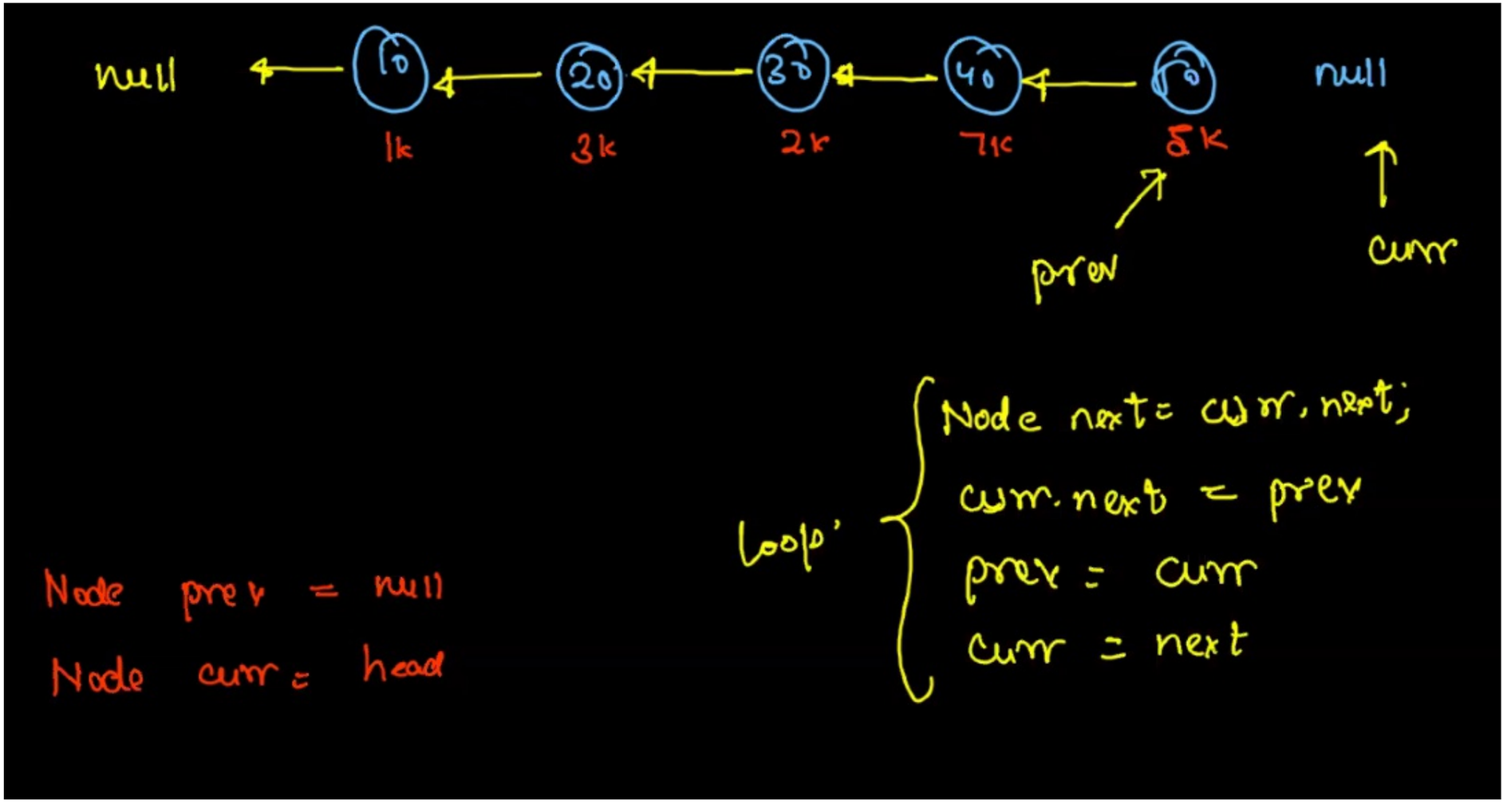
```java
class linkedlist {

    private Node head;
    private Node tail;
    private int size;

    public linkedlist() {
        this.head = this.tail = null;
        this.size = 0;
    }

    private class Node {
        private int data;
        private Node next;

        public Node() {
            this.data = 0;
            this.next = null;
        }

        public Node(int data) {
            this.data = data;
            this.next = null;
        }

        public Node(int data, Node next) {

            this.data = data;
            this.next = next;
        }
    }
}
```



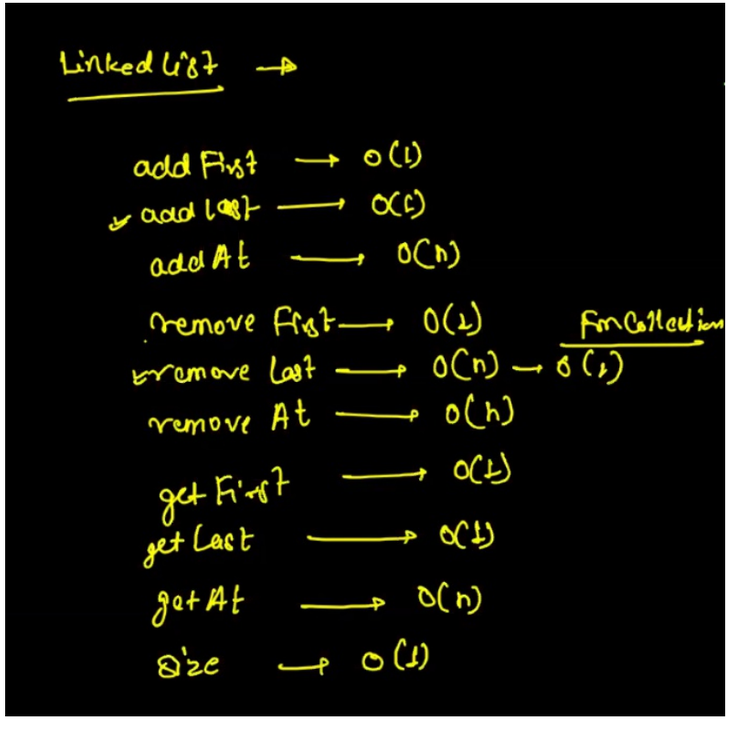
notice : only data reverse and not reference reverse

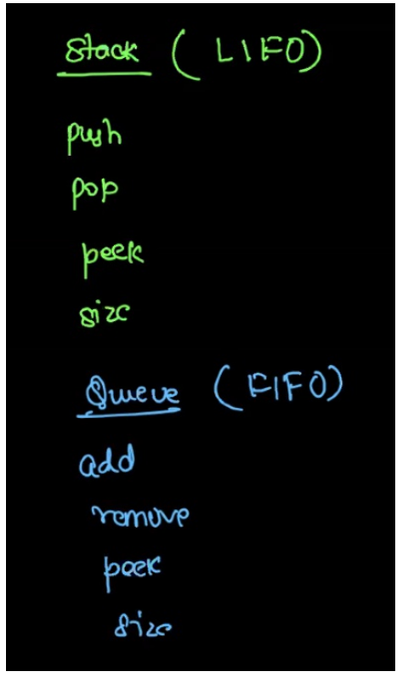
Reverse Linked List (Pointer Iterative)

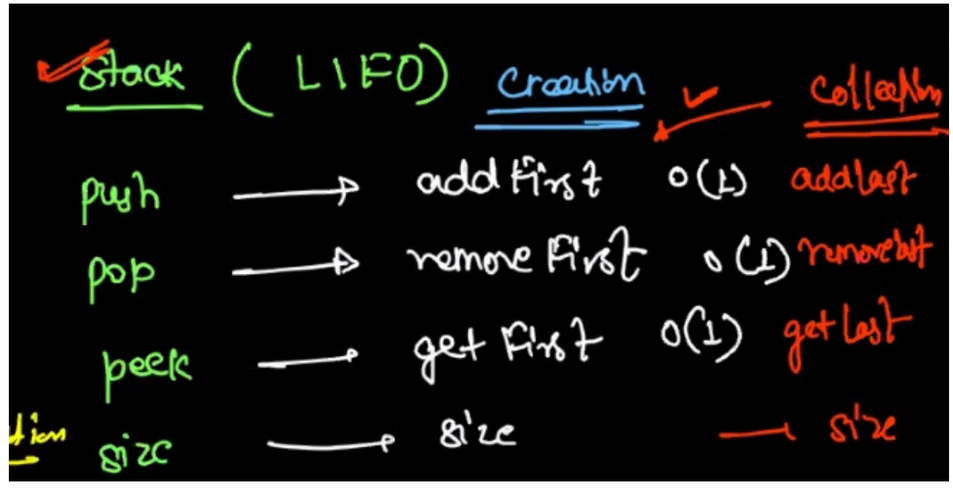



null
10
20
30
40
50
null
1k
3k
2k
71c
5k
prev
curr
Node prev = null
Node curr = head
loop
Node next = curr.next;
curr.next = prev
prev = curr
curr = next

our custom list

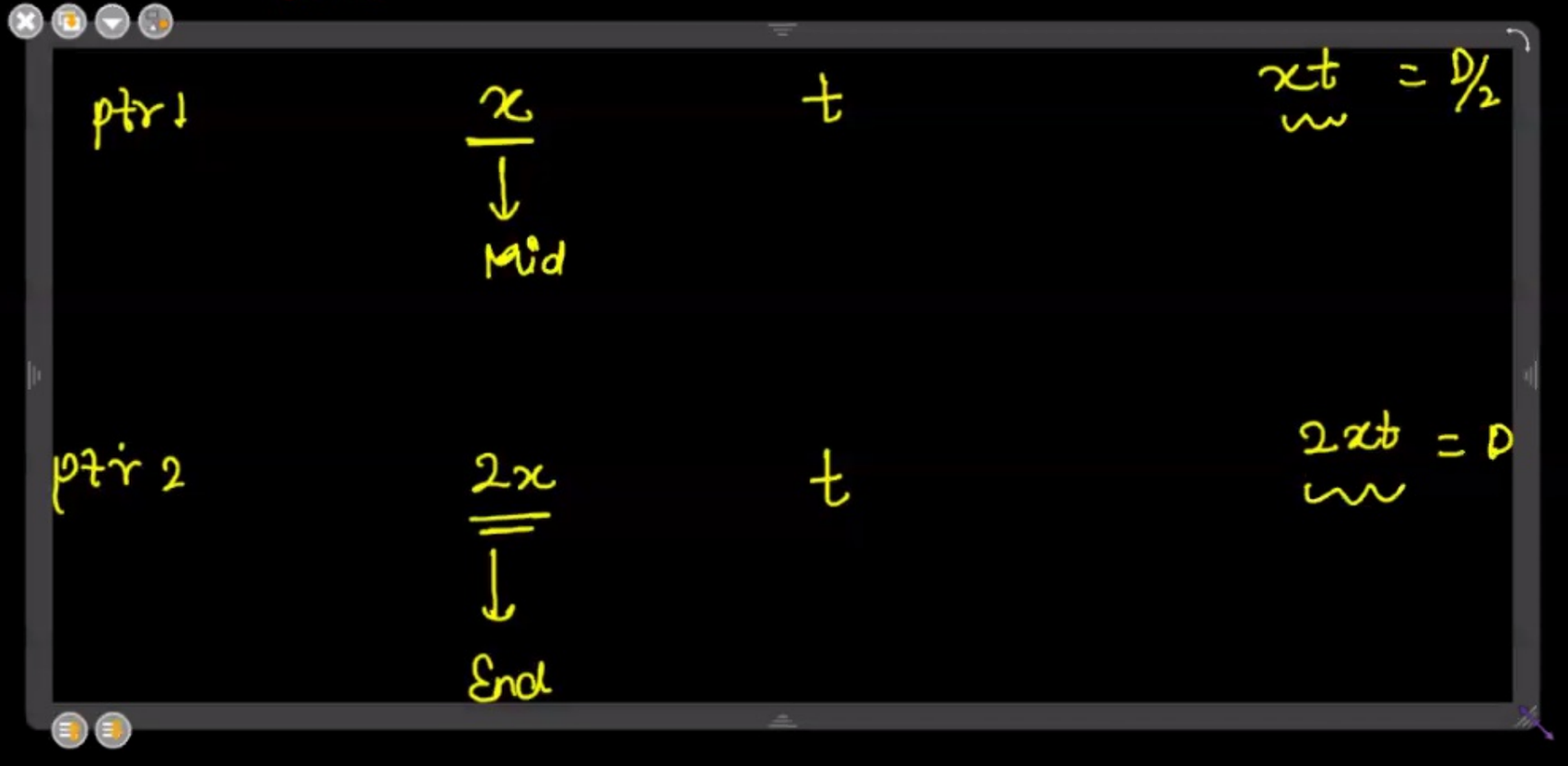






Speed = distance / time

distance = speed × time

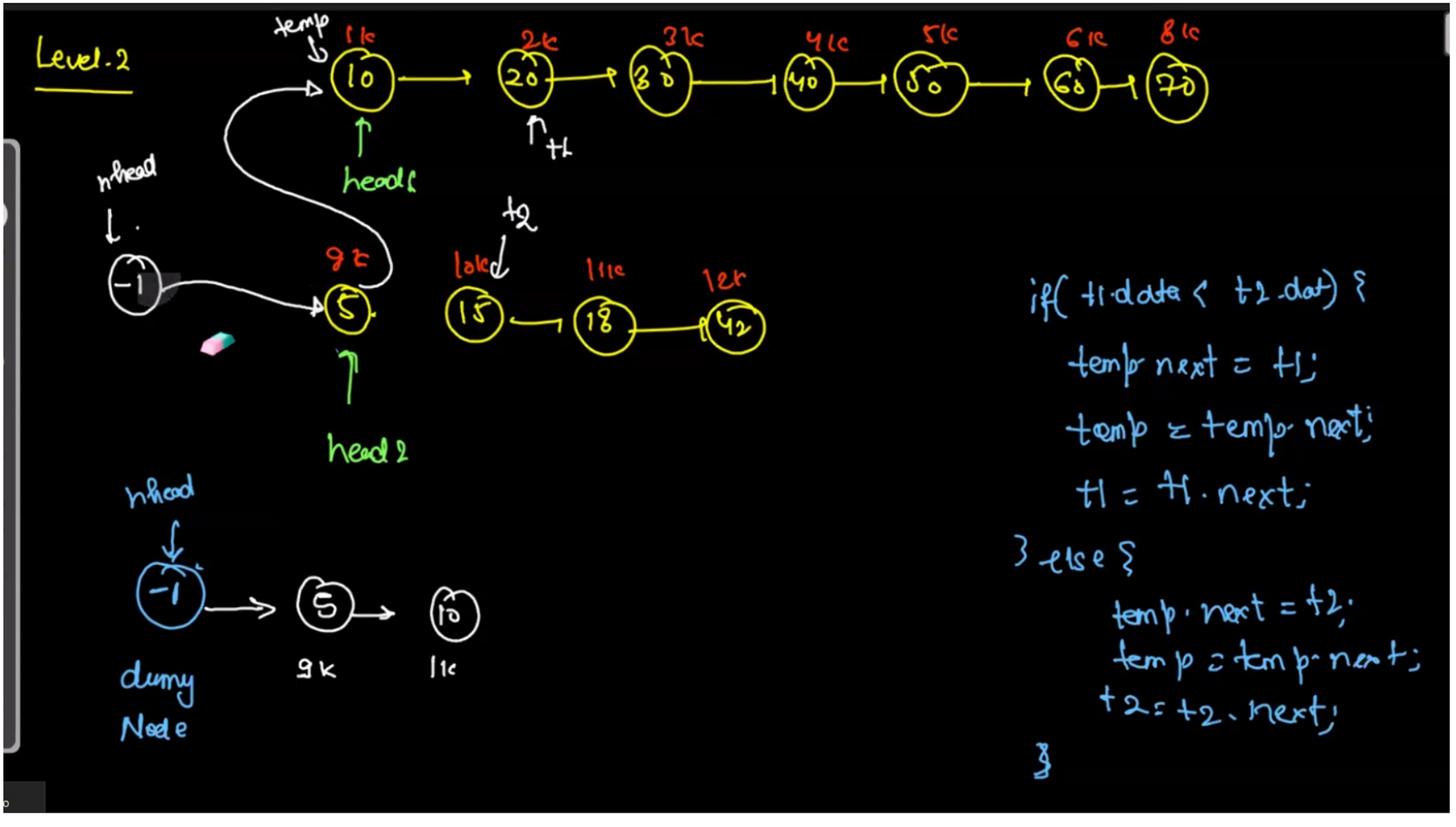


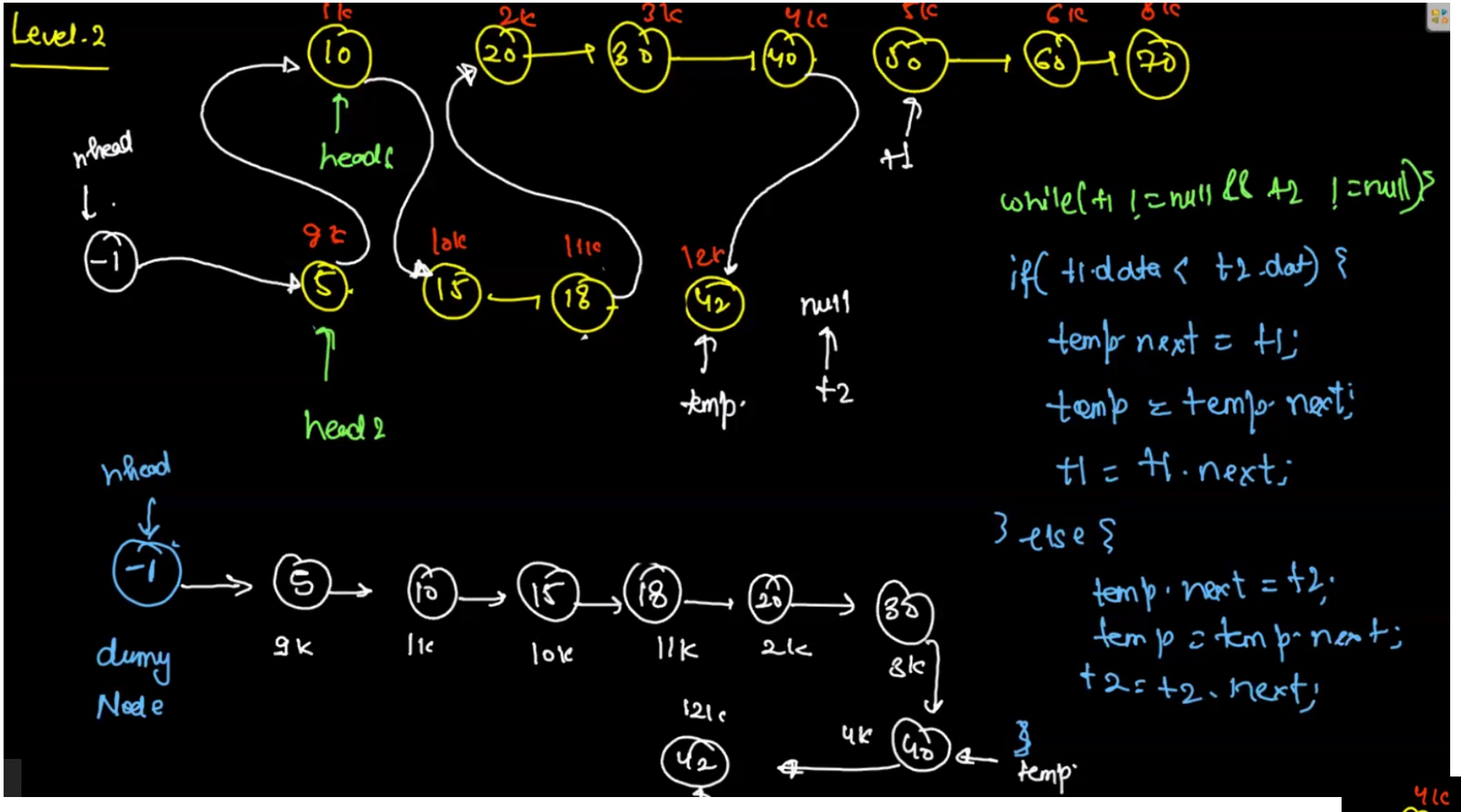
merge two link list with normal approach

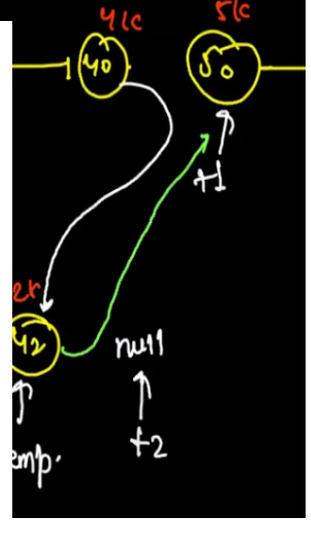
need to make addlast

```
public linkedlist mergeTwoSortedList1(linkedlist l1, linkedlist l2) {
    // changing in given linkedlist is allowed.
    linkedlist res = new linkedlist();
    while(l1.size() > 0 && l2.size() > 0) {
        if(l1.getFirst() < l2.getFirst()) {
            res.addLast(l1.removeFirst());
        } else {
            res.addLast(l2.removeFirst());
        }
    }

    // l1 left over
    while(l1.size() > 0) {
        res.addLast(l1.removeFirst());
    }

    // l2 left over
    while(l2.size() > 0) {
        res.addLast(l2.removeFirst());
    }
    return res;
}
```

[Merge Two Sorted Lists - LeetCode]







when t2 null ho gaya he toh
temp.next =t1

when t1==null
then temp.next=t

```java
public linkedlist mergeTwoSortedList2(linkedlist l1, linkedlist l2) {
    // inplace change in original linkedlist
    Node head1 = l1.head;
    Node head2 = l2.head;

    Node t1 = head1;
    Node t2 = head2;

    Node dummy = new Node(-1);
    Node temp = dummy;

    while(t1 != null && t2 != null) {
        if(t1.data < t2.data) {
            temp.next = t1;
            temp = temp.next;
            t1 = t1.next;
        } else {
            temp.next = t2;
            temp = temp.next;
            t2 = t2.next;
        }
    }
```

```java
if(t1  == null) {
    temp.next = t2;
} else {
    temp.next = t1;
}
```

LeetCode Explore Problems Interview Contest Discuss Store Premium

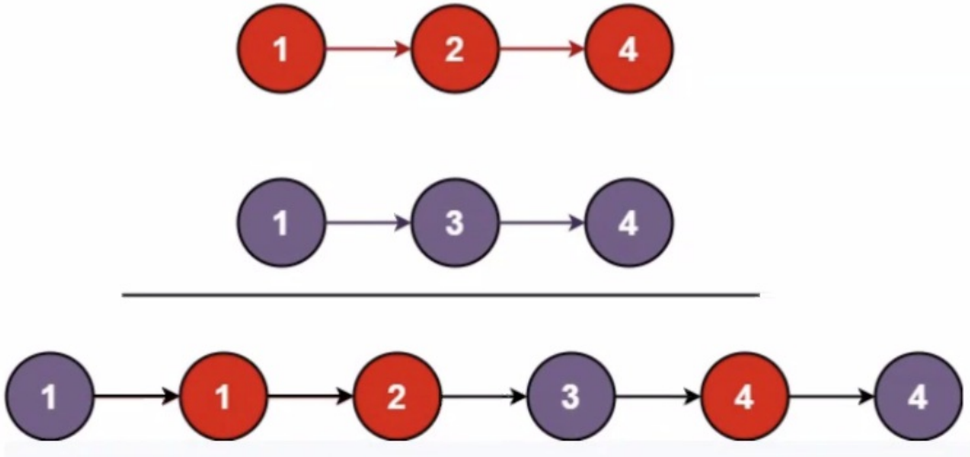
Description Solution Discuss (999+) Submissions

21. Merge Two Sorted Lists

Easy 6786 777 Add to List Share

Merge two sorted linked lists and return it as a **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Example 1:




```
Input: l1 = [1,2,4], l2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
l1 = [], l2 = []
[]
```

to join audio

Java Autocomplete

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode mergeTwoLists(ListNode head1, ListNode head2) {
        ListNode t1 = head1;
        ListNode t2 = head2;

        ListNode dummy = new ListNode(-1);
        ListNode temp = dummy;

        while(t1 != null && t2 != null) {
            if(t1.data < t2.data) {
                temp.next = t1;
                temp = temp.next;
                t1 = t1.next;
            } else {
                temp.next = t2;
                temp = temp.next;
                t2 = t2.next;
            }
        }

        if(t1  == null) {
            temp.next = t2;
        } else {
            temp.next = t1;
        }

        return dummy.next;
    }
}
```

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode mergeTwoLists(ListNode head1, ListNode head2) {
        ListNode t1 = head1;
        ListNode t2 = head2;

        ListNode dummy = new ListNode(-1);
        ListNode temp = dummy;

        while(t1 != null && t2 != null) {
            if(t1.val < t2.val) {
                temp.next = t1;
                temp = temp.next;
                t1 = t1.next;
            } else {
                temp.next = t2;
                temp = temp.next;
                t2 = t2.next;
            }
        }

        if(t1  == null) {
            temp.next = t2;
        } else {
            temp.next = t1;
        }

        return dummy.next;
    }
}
```