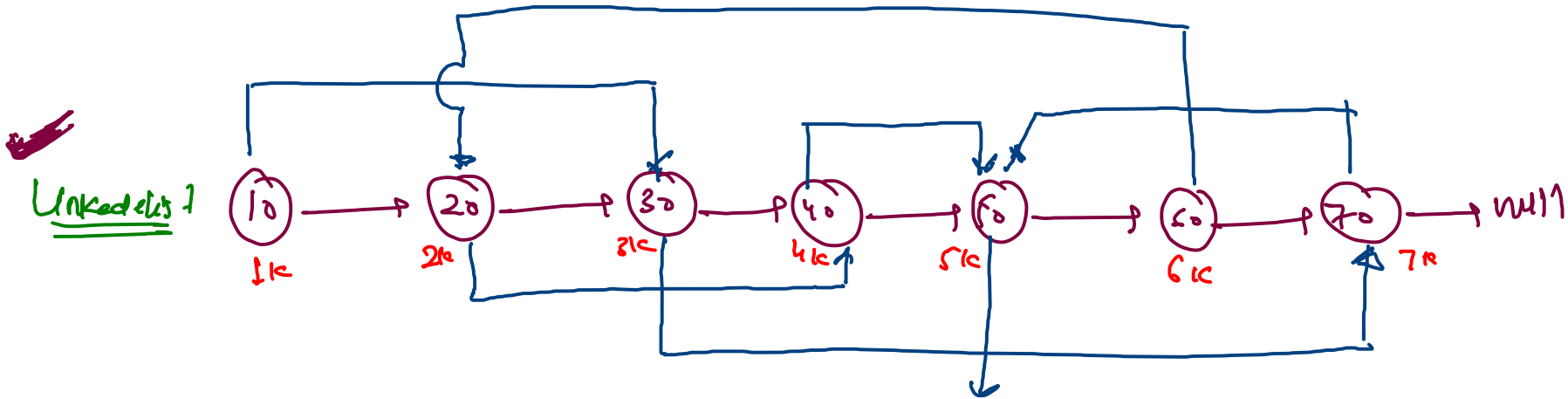
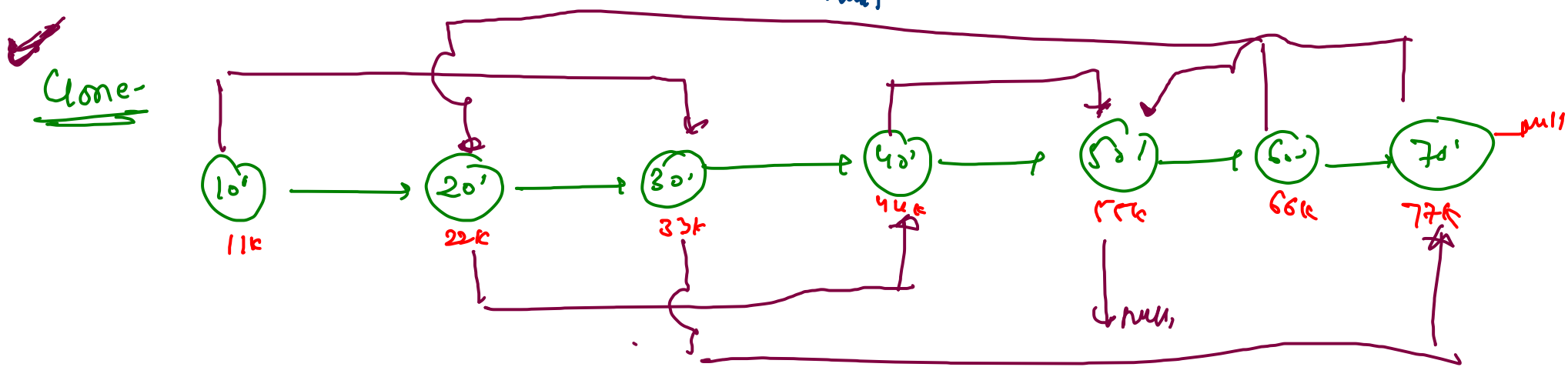
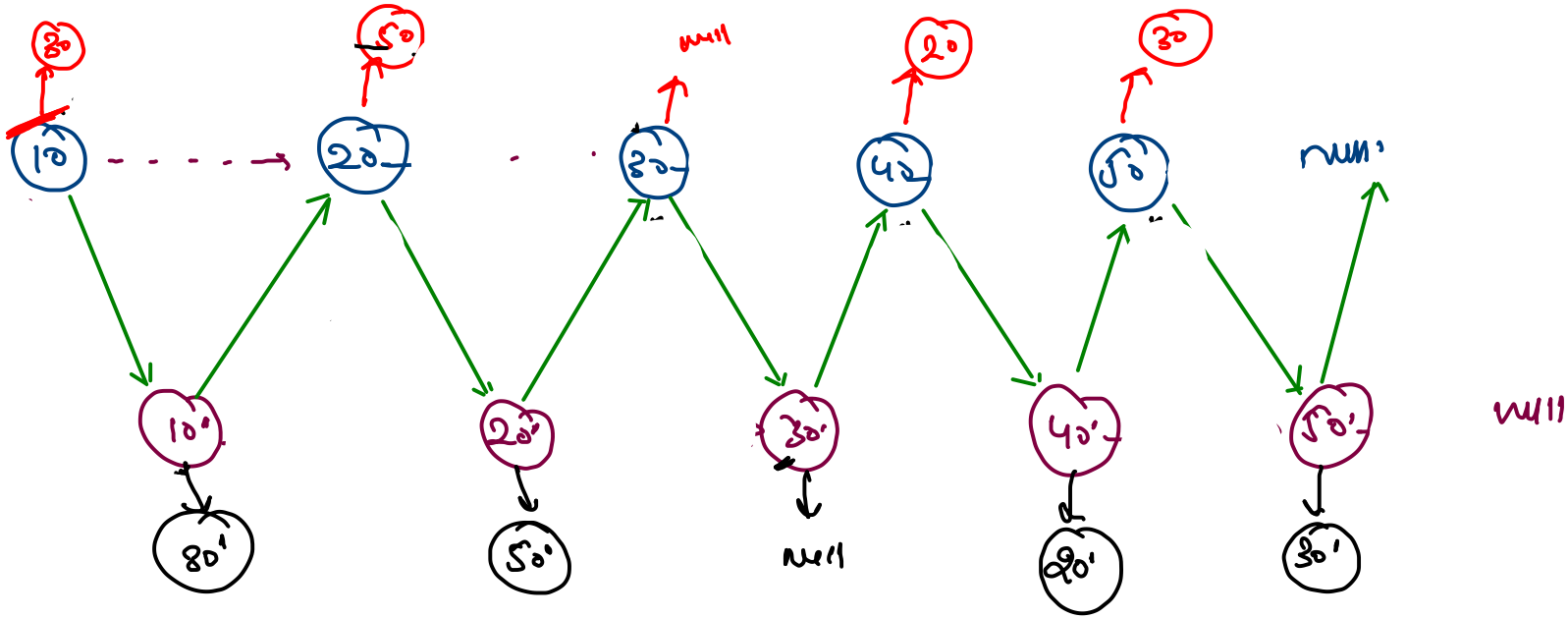
copy LinkedList with Random pointer:

Class → int data;
Node next;
Node random







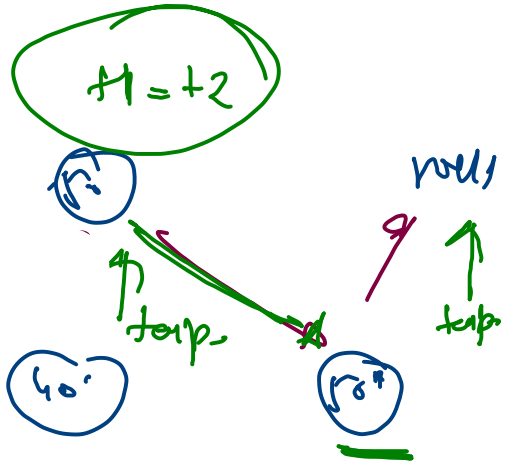
Set

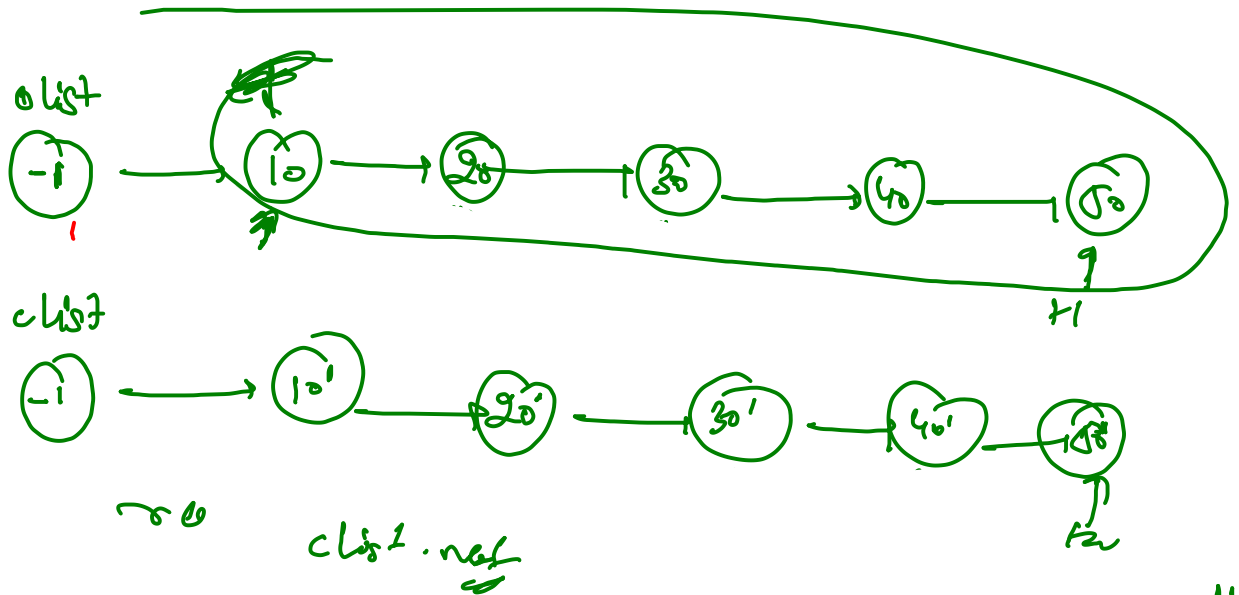
temp.next.random = temp.random == null ? null : temp.random.next;

forward

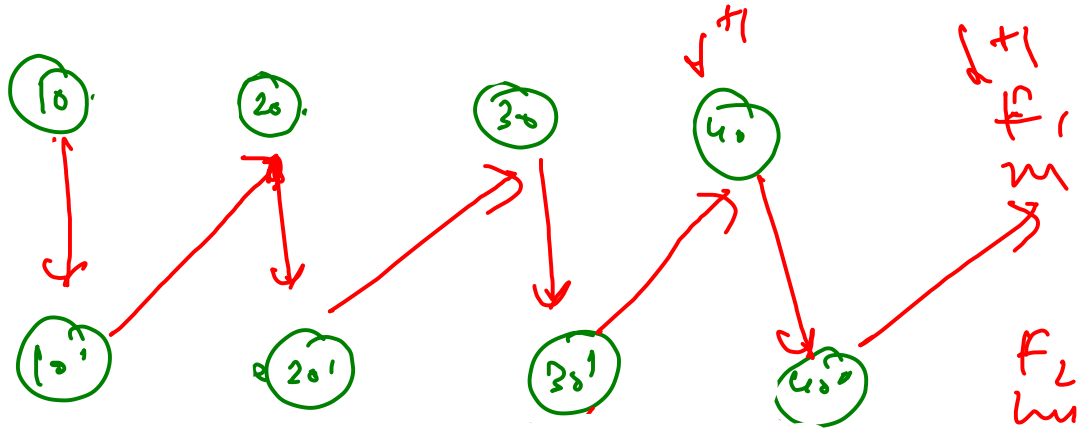
temp = temp.next.next.

1. Make normal clone list
2. Alternate arrangement
3. Make Random pointers of clone
4. LinkedList Recover o.list.






```
t1.next = temp;
t1 = temp;
t2.next = temp.next;
t2 = temp.next
temp = temp.next.next;
t1.next = null;
```



forward1 : t1.next;

forward2 : t2.next)

t1.next : t2;

t2.next = forward1.

t1 = f1;

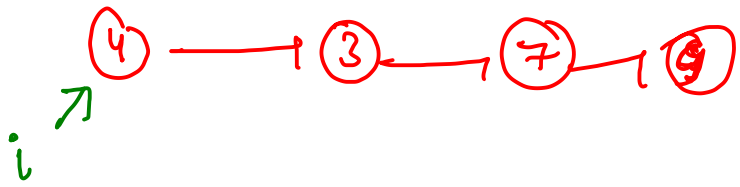
t2 = f2;

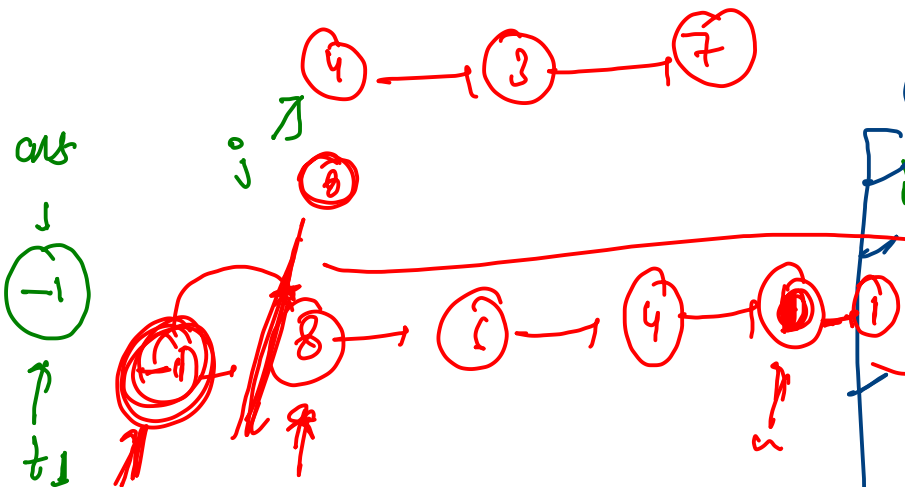
Add two Linked List →

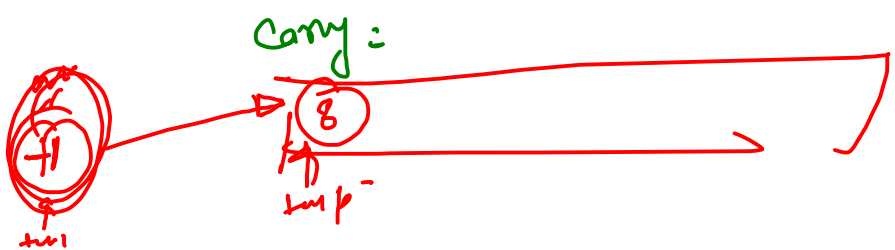
9 7 3 4

7 3 4

1 0 4 6 8







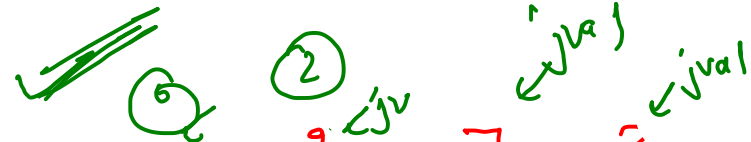

```
(i!=null || j!=null || carry==1)
ival = i==null ? 0 : i.val;
i = i==null ? null : i.next;
jval = j==null ? 0 : j.val;
j = j==null ? null : j.next;
Sum = ival + jval + carry
val = Sum % 10;
carry = Sum / 10;
nn = new ListNode(val);
{ t1.next = nn;
  t1 = nn;
```

Step

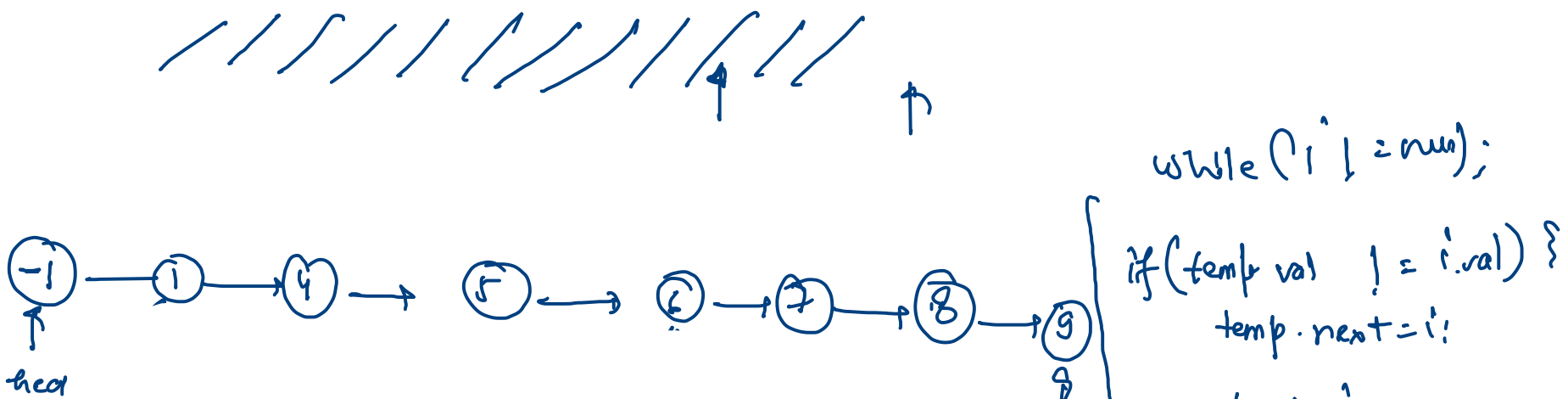
(1) Reverse

Subtract two LinkedList →

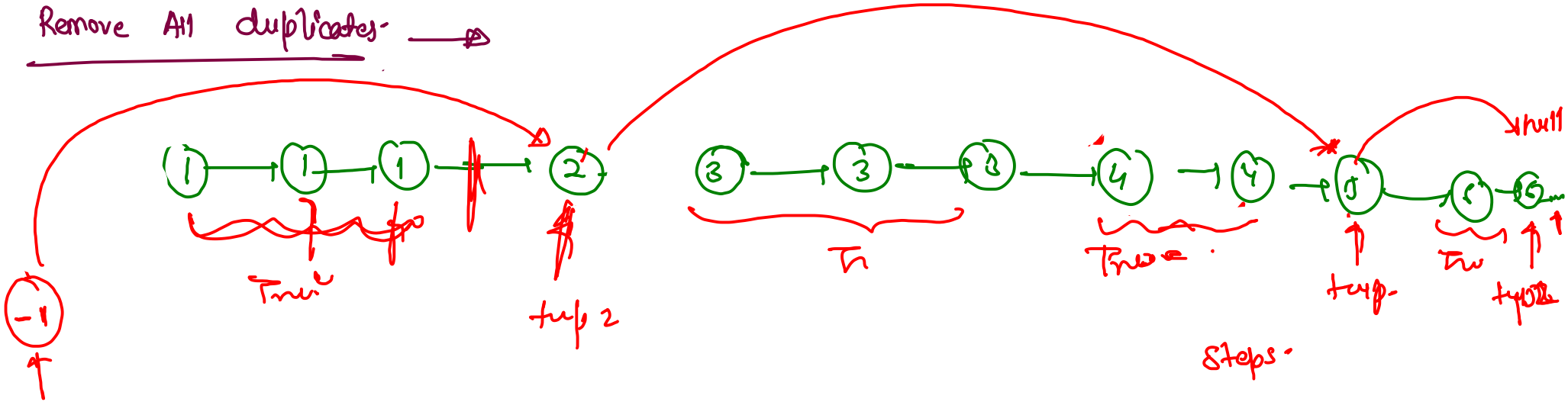
Multiplication of two linked list



Remove duplicate from Sorted List



Remove All duplicates.



Steps.

dummy.

temp = dum

temp next = head.

temp2 = head.next

```java
public static ListNode removeDuplicates2(ListNode head) {
    if(head == null || head.next == null) return head;
    ListNode nhead = new ListNode(-1);
    ListNode temp = nhead, temp2 = head.next;

    temp.next = head;

    while(temp2 != null) {
        boolean flag = false;
        while(temp2 != null && temp.next.val == temp2.val) {
            flag = true;
            temp2 = temp2.next;
        }

        if(flag == true) {
            temp.next = temp2;
        } else {
            temp = temp.next;
        }

        if(temp2 != null)
            temp2 = temp2.next;
    }
    return nhead.next;
}
```