c1

$l_1 =$ $1 \leftarrow 2^1 \leftarrow 9^2 \leftarrow 3^1 \leftarrow 7$

c2

$l_2 =$ $8 \leftarrow 6 \leftarrow 4$

0 to 9

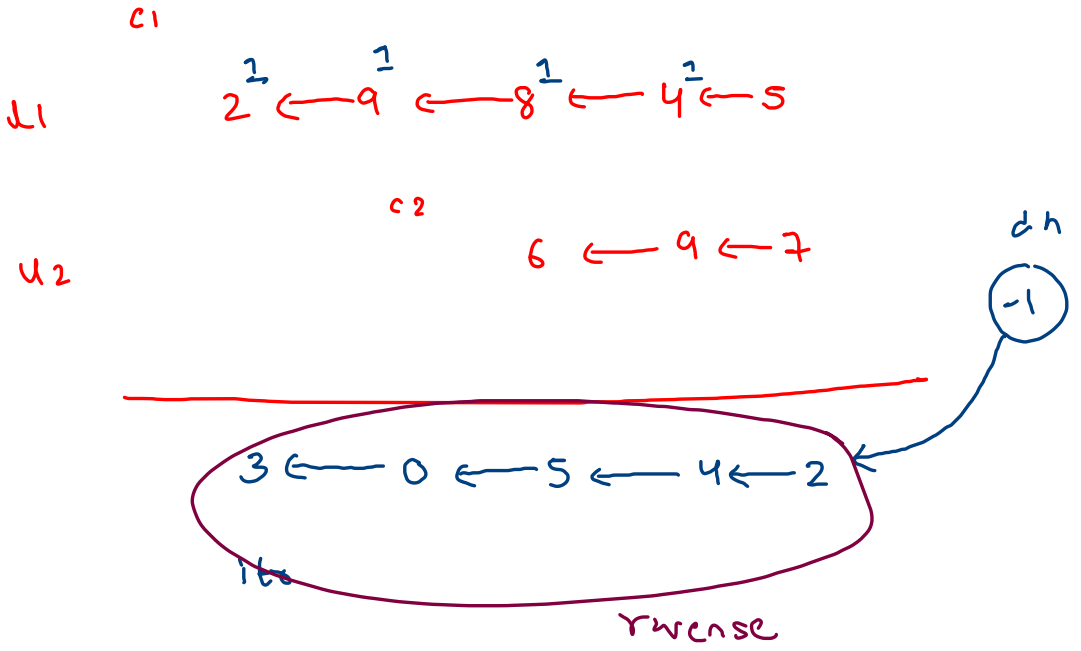
n

-1

$1 \leftarrow 3 \leftarrow 8 \leftarrow 0 \leftarrow 1$

itr

```java
public static ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    ListNode dh = new ListNode(-1);
    ListNode itr = dh;

    ListNode c1 = reverse(l1);
    ListNode c2 = reverse(l2);

    int carry = 0;

    while(c1 != null || c2 != null || carry != 0) {
        int sum = carry;

        if(c1 != null) {
            sum += c1.val;
            c1 = c1.next;
        }

        if(c2 != null) {
            sum += c2.val;
            c2 = c2.next;
        }

        int val = sum % 10;
        carry = sum / 10;

        ListNode nn = new ListNode(val);
        itr.next = nn;

        itr = itr.next;
    }

    return reverse(dh.next);
}
```



Sum = 1 + 2

val = 0

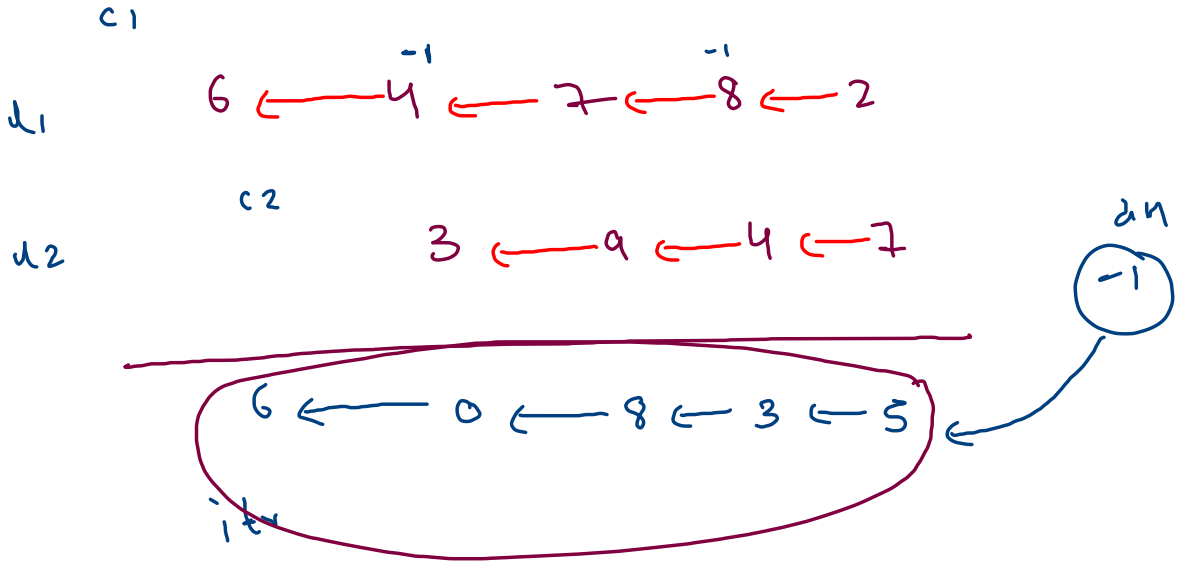
C = 0

29845

+ 697

Ans = 3 -> 0 -> 5 -> 4 -> 2

Subtraction

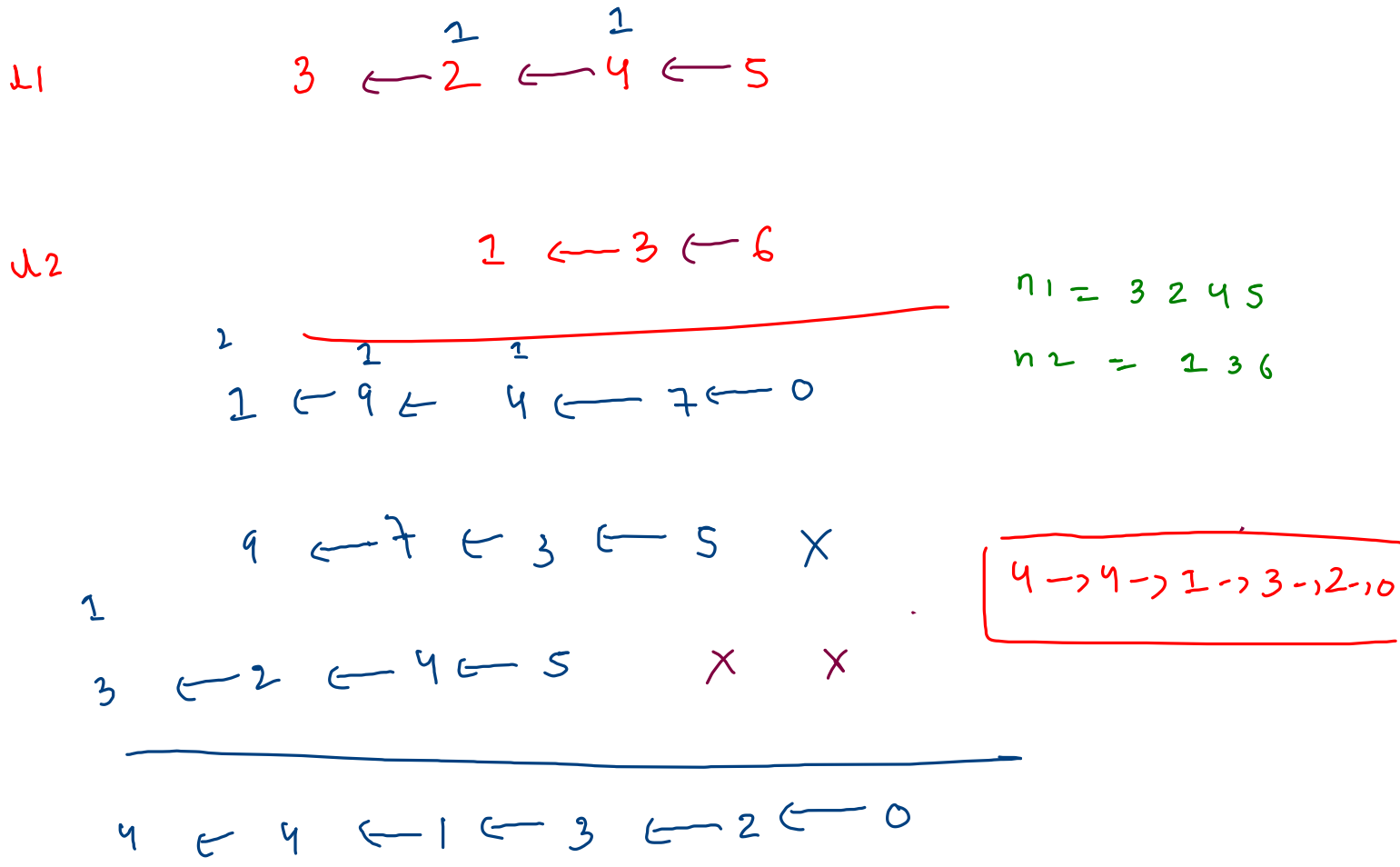


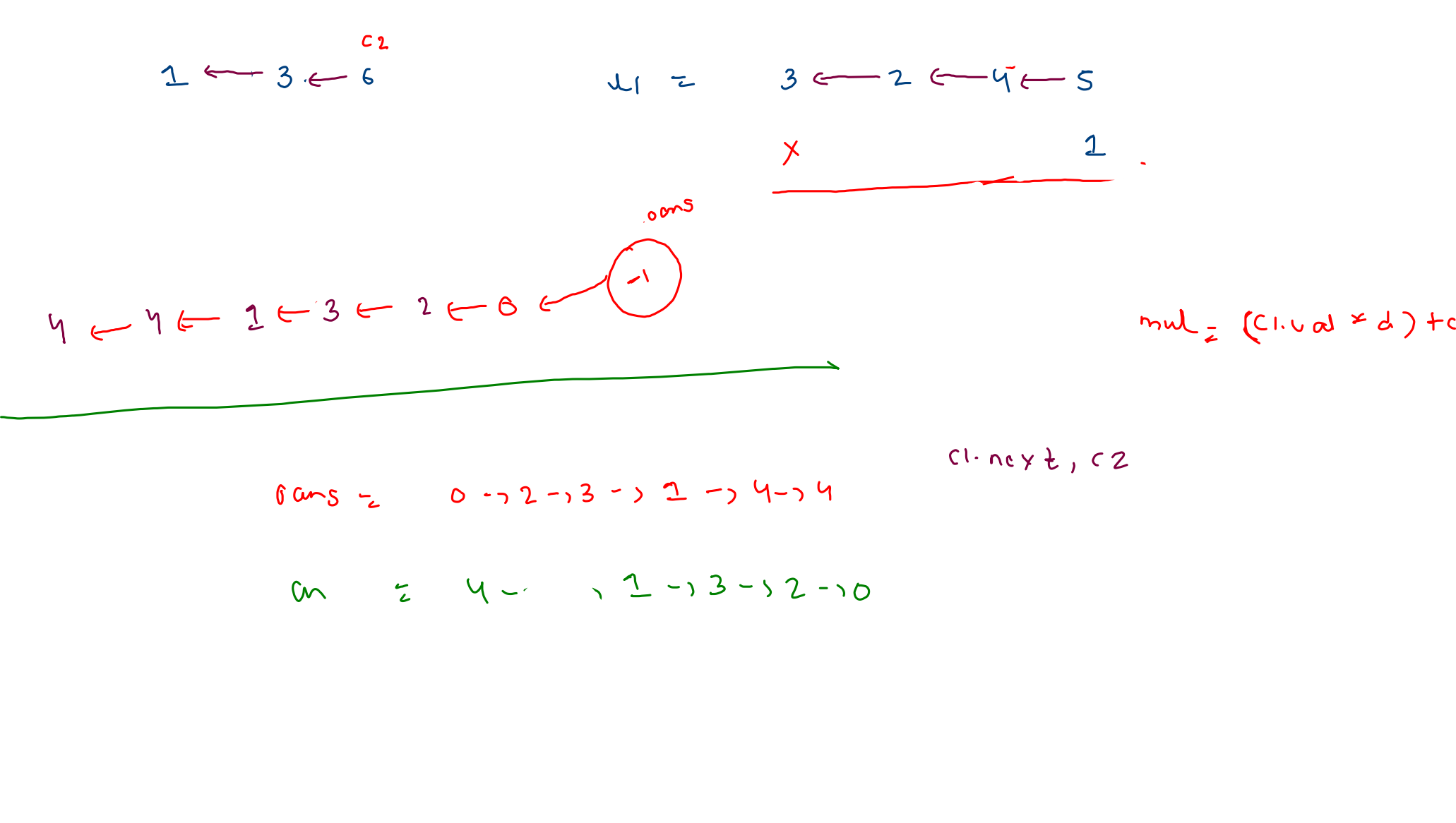
dh.next = 5 -> 3 -> 8 -> 0 -> 6

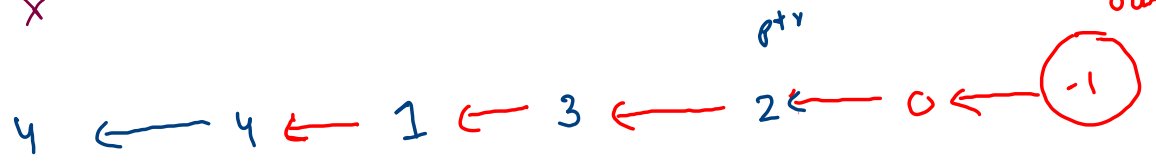
ans = 6 -> 0 -> 8 -> 3 -> 5

6 4 7 8 2

- 3 9 4 7


l1
3 ← 2 ← 4 ← 5
1 1
l2
1 ← 3 ← 6
n1 = 3 2 4 5
n2 = 1 3 6
2 1 1
1 ← 9 ← 4 ← 7 ← 0
9 ← 7 ← 3 ← 5 X
1
3 ← 2 ← 4 ← 5 X X
4 ← 4 ← 1 ← 3 ← 2 ← 0
4 -> 4 -> 1 -> 3 -> 2 -> 0


c2
1 ← 3 ← 6
l1 = 3 ← 2 ← 4 ← 5
X 1
ans
-1
4 ← 4 ← 1 ← 3 ← 2 ← 0
mul = (c1.val * c
c1.next, c2
ans = 0 -> 2 -> 3 -> 1 -> 4 -> 4
ans = 4 ... , 1 -> 3 -> 2 -> 0



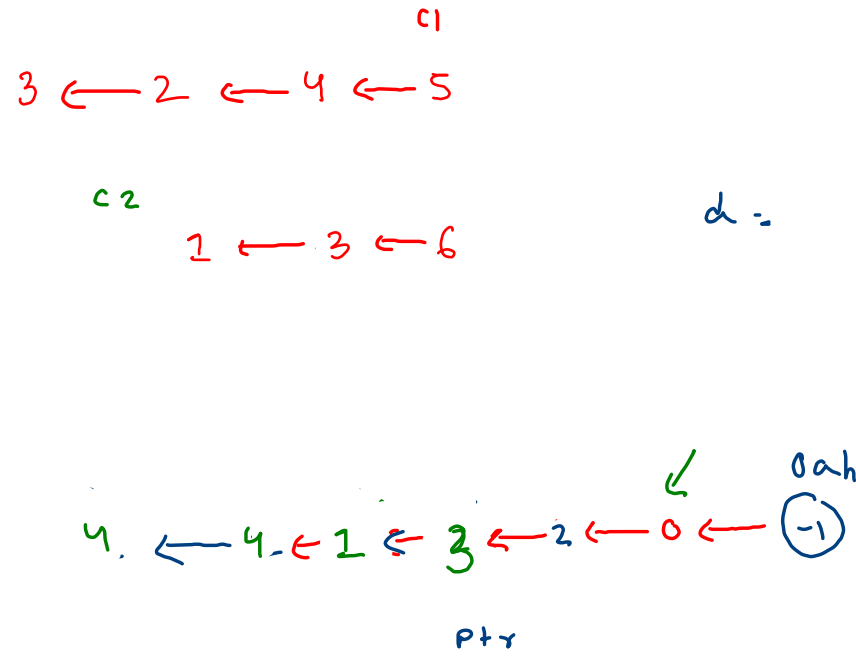
c1.next, c2

4 -> 4 -> 1 -> 3 -> 2 -> 0

```java
public static ListNode multiplyTwoLL(ListNode l1, ListNode l2) {
    ListNode oah = new ListNode(-1);
    ListNode ptr = oah; //overall answer head

    ListNode c1 = reverse(l1);
    ListNode c2 = reverse(l2);

    while(c2 != null) {
        int d = c2.val;
        c2 = c2.next;

        ListNode spdll = singleDigitWithLLMult(c1,d); //single pointer

        addTwoLL(spdll,ptr);

        ptr = ptr.next;
    }

    return reverse(oah.next);
}
```

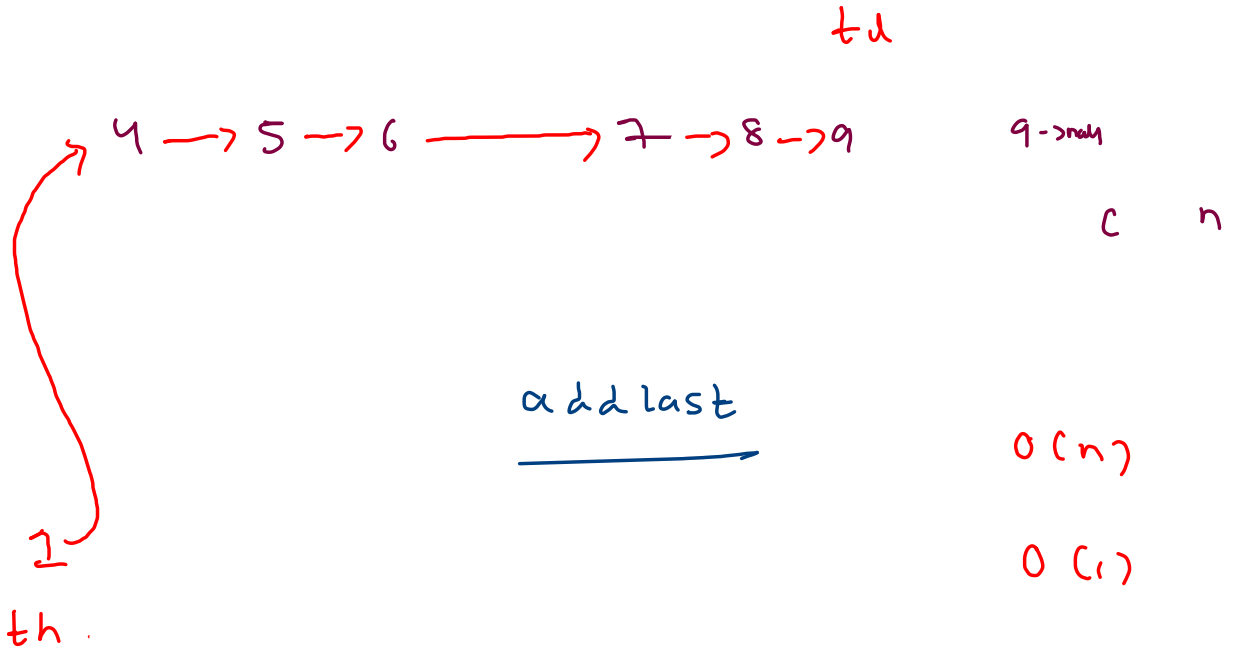
```java
public static void addTwoLL(ListNode spdll,ListNode ptr) {
    ListNode c2 = spdll;
    ListNode c1 = ptr;

    int carry = 0;

    //c1.next and c2 will interact
    while(c1.next != null || c2 != null || carry != 0) {
        int sum = carry;

        if(c2 != null) {
            sum += c2.val;
            c2 = c2.next;
        }

        if(c1.next != null) {
            sum += c1.next.val;
        }
        else {
            c1.next = new ListNode(0);
        }

        int val = sum % 10;
        carry = sum / 10;

        c1.next.val = val;
        c1 = c1.next;
    }

}
```

```java
public static ListNode singleDigitWithLLMult(ListNode l1,int d) {
    ListNode dh = new ListNode(-1);
    ListNode itr = dh;

    ListNode c1 = l1;
    int carry = 0;

    while(c1 != null || carry != 0) {
        int mult = carry;

        if(c1 != null) {
            mult += c1.val*d;
            c1 = c1.next;
        }

        int val = mult % 10;
        carry = mult / 10;

        itr.next = new ListNode(val);
        itr = itr.next;
    }

    return dh.next;

}
```

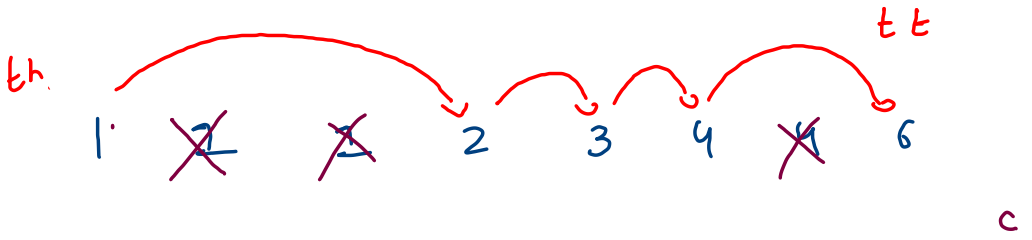


ans → 4 → 4 → 1 → 3 → 2 → 0

1->1->1->4->5->6->6->7->8->9->9->9->null



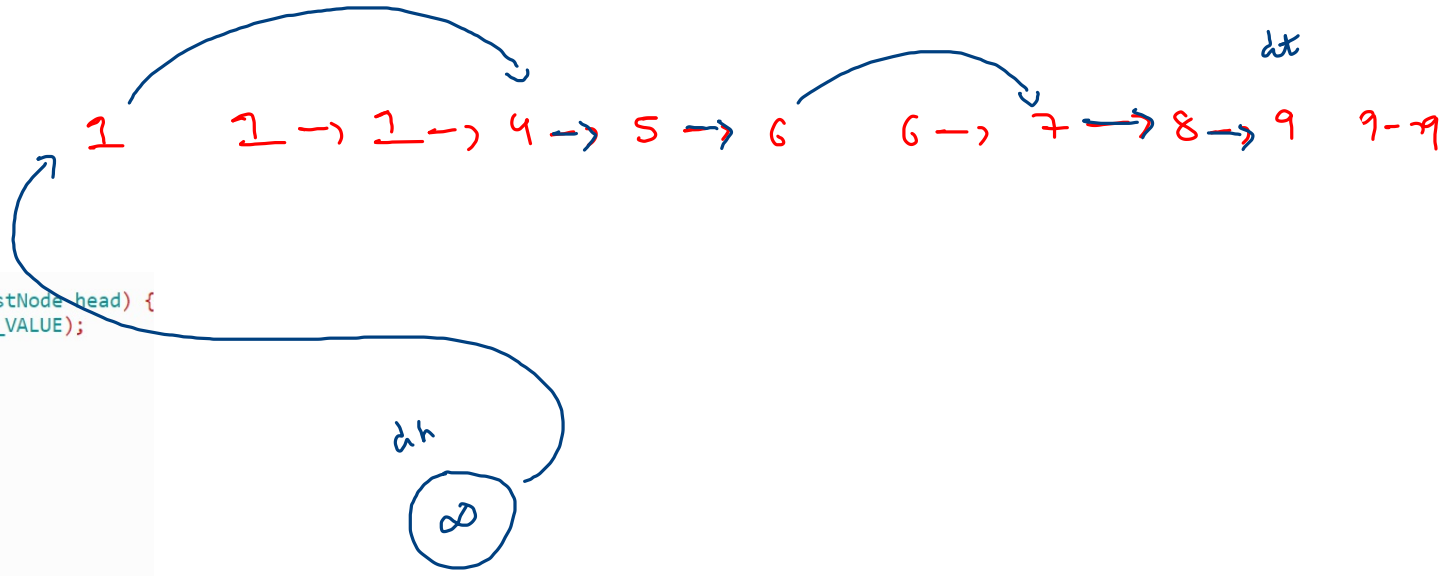
```java
public static void addLast(ListNode node) {
    if(th == null) {
        th = tt = node;
    }
    else {
        tt.next = node;
        tt = tt.next;
    }
}

public static ListNode removeDuplicates(ListNode head) {

    ListNode curr = head;

    while(curr != null) {

        ListNode next = curr.next;
        curr.next = null;

        if(tt == null || tt.val != curr.val) {
            addLast(curr);
        }

        curr = next;
    }

    return th;
}
```



time O(n)

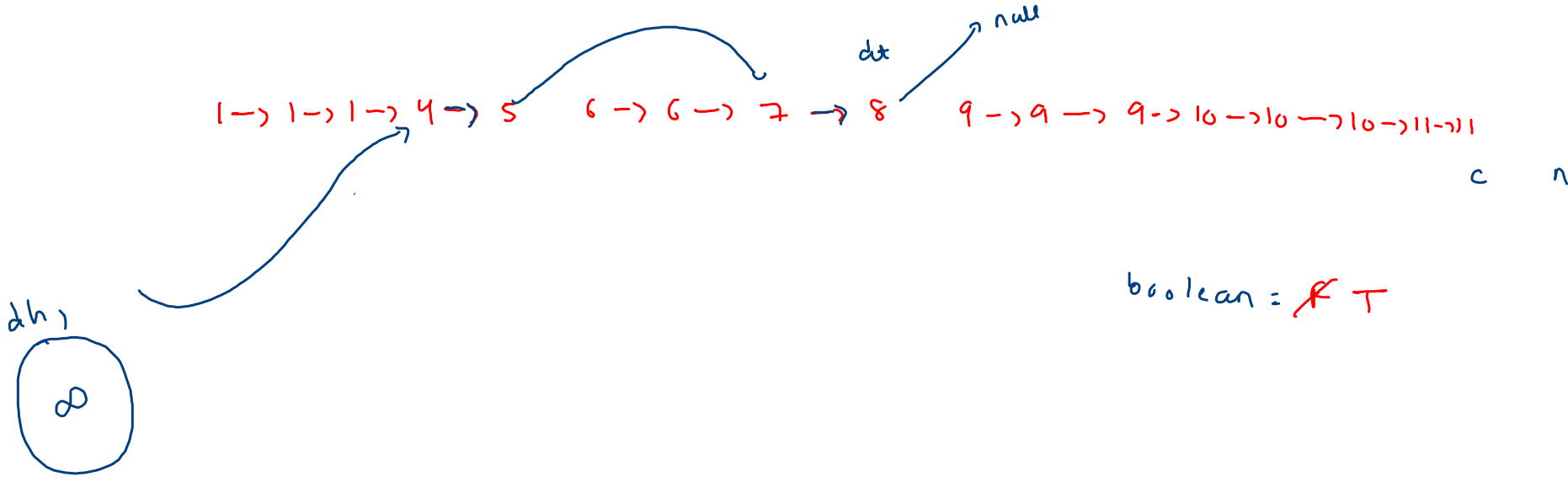
space O(1)

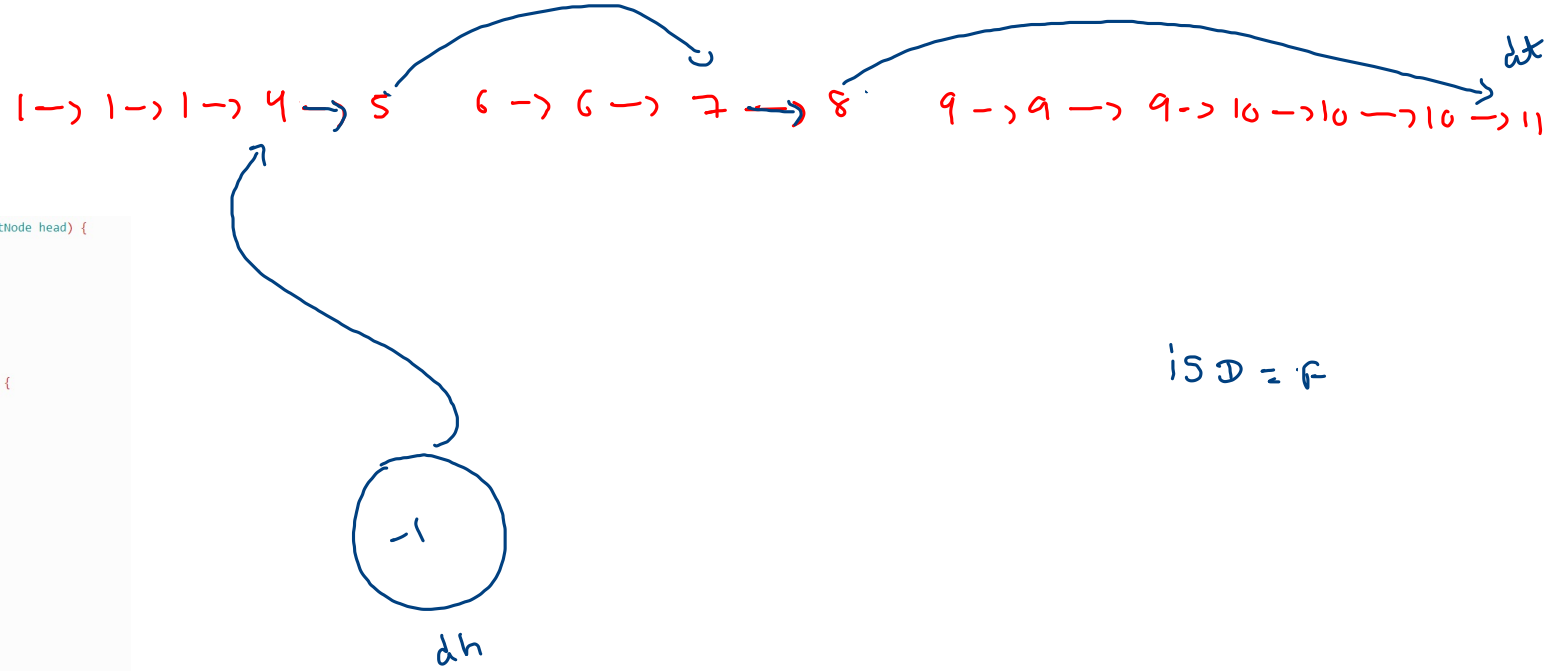
1->1->1->4->5->6->6->7->8->9->9->9->null



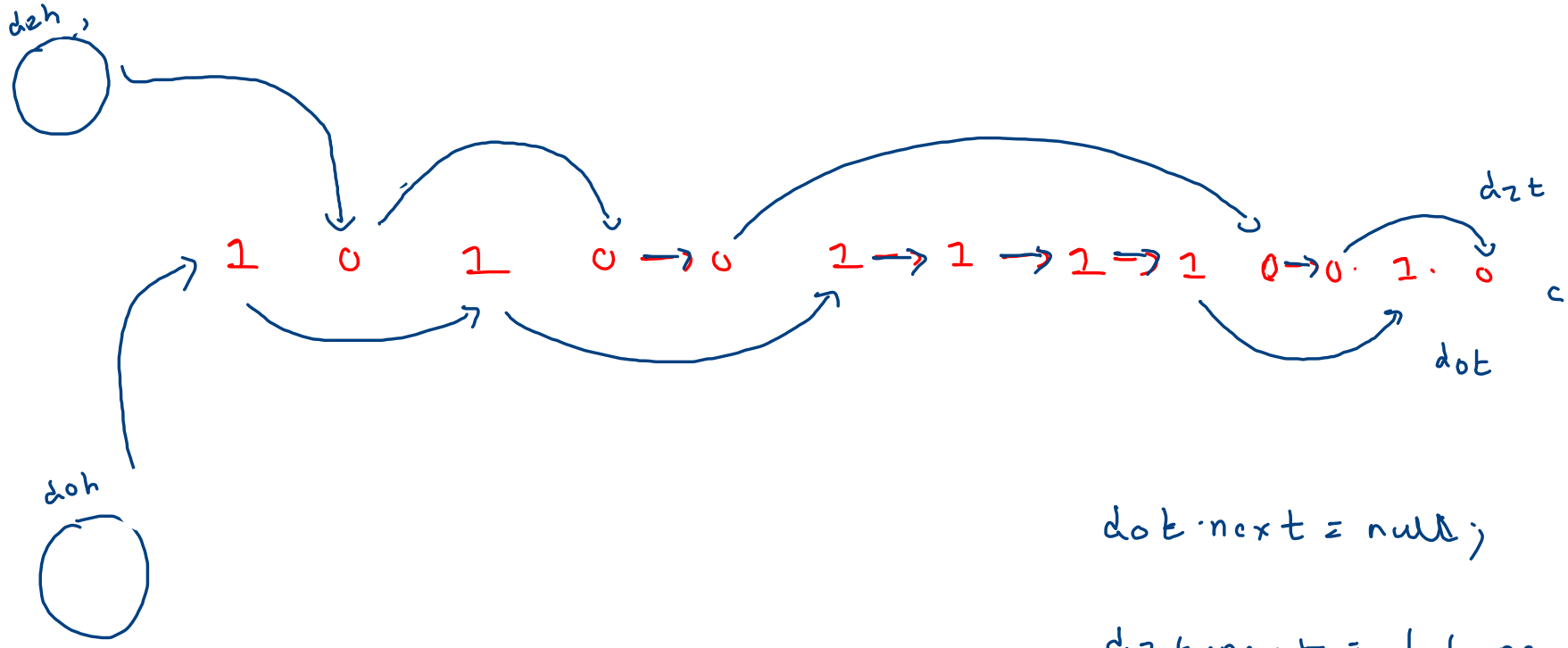

```java
public static ListNode removeDuplicates(ListNode head) {
    ListNode dh = new ListNode(Integer.MAX_VALUE);
    ListNode dt = dh;

    ListNode curr = head;

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

    dt.next = null;

    return dh.next;
}
```

1->1->1->4->5->6->6->7->8->9->9->9->null






```java
public static ListNode removeDuplicates(ListNode head) {
    ListNode dh = new ListNode(-1);
    ListNode dt = dh;
    ListNode curr = head;
    boolean isDup = false;

    while(curr != null) {
        isDup = false;

        ListNode next = curr.next;
        while(curr != null && next != null) {
            if(curr.val != next.val) {
                break;
            }
            isDup = true;
            curr = next;
            next = next.next;
        }

        if(isDup == false) {
            //we should use curr
            dt.next = curr;
            dt = dt.next;
        }

        curr = next;
    }

    if(isDup == false) {
        //we should use curr
        dt.next = curr;
        dt = dt.next;
    }
    else {
        dt.next = null;
    }

    return dh.next;
}
```



d1t.next = null;

d2t.next = d1h.next;

ans -> d2h.next

```java
public static ListNode segregate01(ListNode head) {
    ListNode fo =  null;//first one
    ListNode curr = head;

    while(curr != null) {
        if(curr.val == 1) {
            if(fo == null) {
                //this is first one
                fo = curr;
            }
        }
        else {
            if(fo != null) {
                //swap curr(0) with first one node(1)
                fo.val = 0;
                curr.val = 1;
                fo = fo.next;
            }
        }
        curr = curr.next;
    }

    return head;
}
```