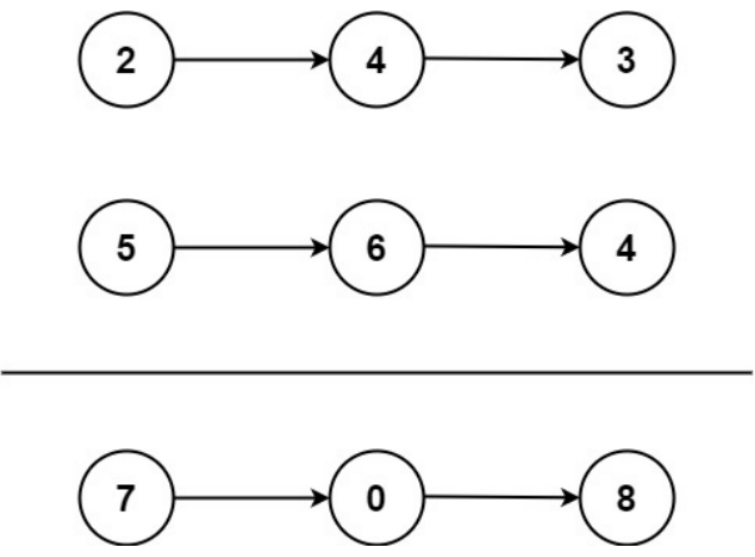
<https://leetcode.com/problems/add-two-numbers/description/>

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**



Input: l1 = [2,4,3], l2 = [5,6,4]

Output: [7,0,8]

Explanation: 342 + 465 = 807.

**Example 2:**

Input: l1 = [0], l2 = [0]

Output: [0]

**Example 3:**

Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]

Output: [8,9,9,9,0,0,0,1]

**Constraints:**

* The number of nodes in each linked list is in the range [1, 100].
* 0 <= Node.val <= 9
* It is guaranteed that the list represents a number that does not have leading zeros.

**Attempt 1: 2023-02-18**

**Solution 1:  Iterative Solution (30 min)**

/\*\*

\* 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 addTwoNumbers(ListNode l1, ListNode l2) {

ListNode iter1 = l1;

ListNode iter2 = l2;

int sum = 0;

ListNode dummy = new ListNode();

ListNode iter = dummy;

while(iter1 != null || iter2 != null) {

sum /= 10;

if(iter1 != null) {

sum += iter1.val;

iter1 = iter1.next;

}

if(iter2 != null) {

sum += iter2.val;

iter2 = iter2.next;

}

iter.next = new ListNode(sum % 10);

iter = iter.next;

}

if(sum / 10 > 0) {

iter.next = new ListNode(1);

}

return dummy.next;

}

}

Time Complexity: O(n)

Space Complexity: O(n)

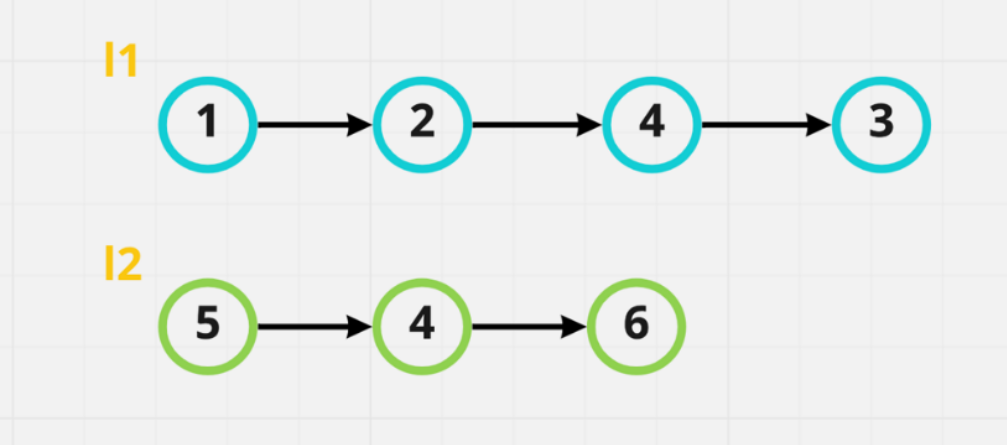
**Refer to**

<https://leetcode.com/problems/add-two-numbers/solutions/1835535/java-c-a-very-beautiful-explanation-ever-exists/>

How's going, Ladies n Gentlemen, today we are going to solve one of the coolest problem **Add Two Numbers**

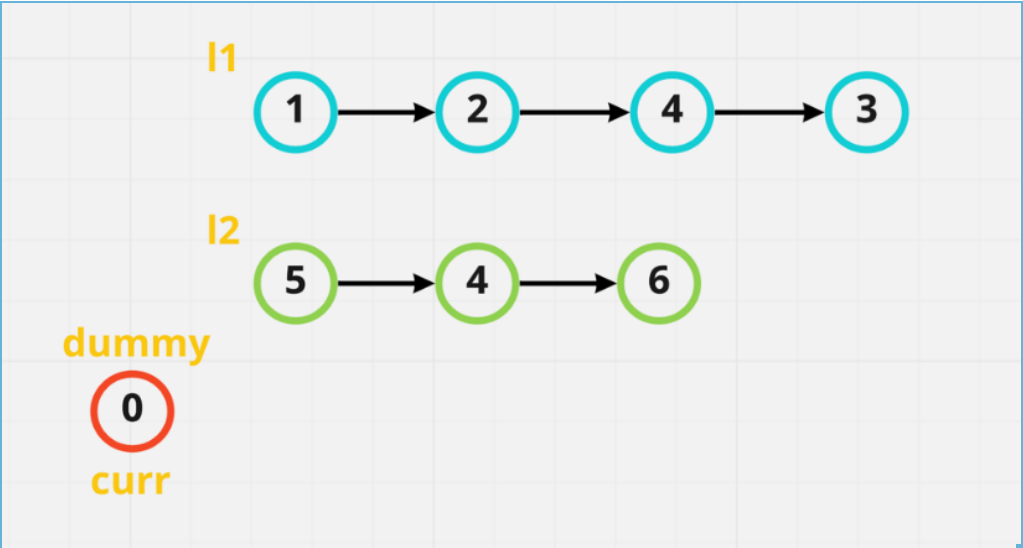
So, what the problem statement is saying we have given 2 linkedlist we have to add them and get the sum in another linkedlist.

What, am saying let's understand with an example:-**Input**: l1 = [1,2,4,3], l2 = [5,4,6]**Output**: [6,6,0,4]



Now let's create another list in which we will get our sum. So, that list initially we will called as dummy list with any value of your choice present in that. *I'll put 0 as we indian has invented that. <^^>*

And one more last thing, we'll gonna create one pointer and let's say i'll call it **curr** which is pointing on dummy node and traverse along with it



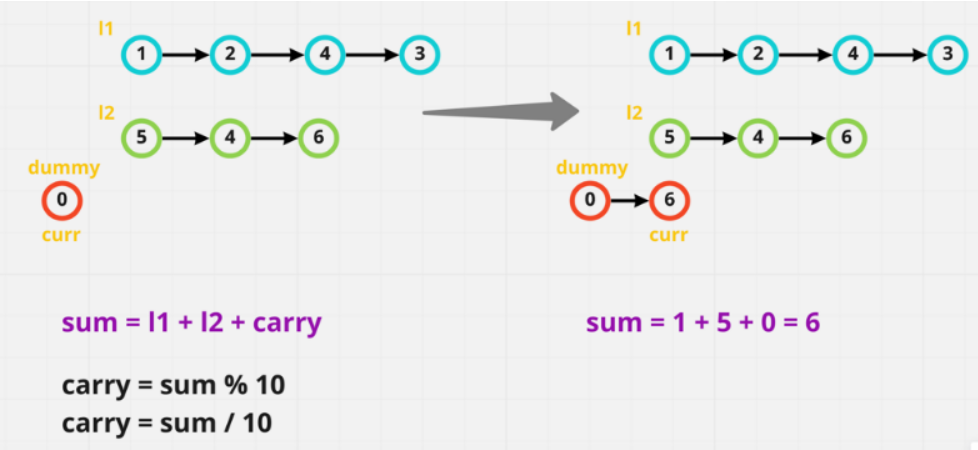
Alright so, here we go ladies n gentlemen, It's time to sum up these node value, for that we will create one another variable let's called it **sum** and put the sum of **l1 & l2** them into our dummy list. So, we start it from all the way left go to all the way right. Now you will ask, dude what about the carry values we get after sum up. Well, hold on i'm coming on that point don't worry.

So, for that what you have to do is, we will initialize one more variable name **carry** if we found carry of let's say 10. First we will modulo it like carry = sum % 10 i.e. carry = 10 % 10 i.e. 0 we will add 0 into our node and after that what we will do is get the carry as carry = sum / 10 i.e. carry = 10 / 10 i.e. 1. Now we are having carry as 1. So, in the next node sum of l1 & l2 we will add carry as well.

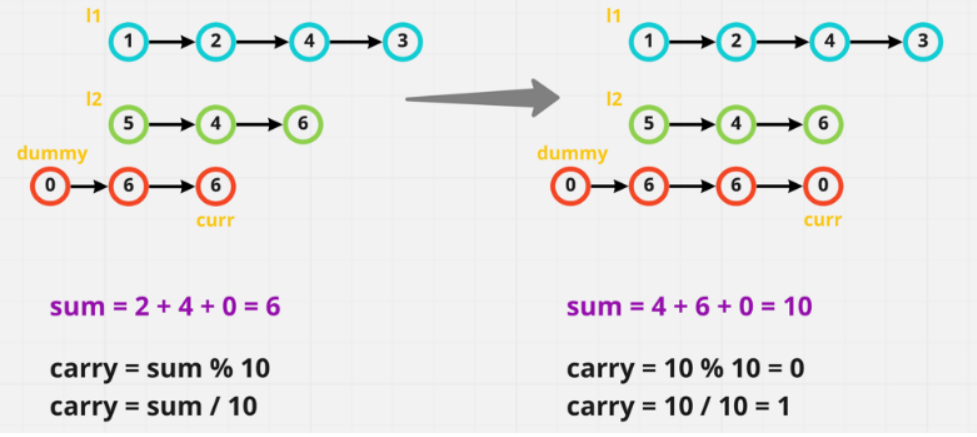
For sum we will use this formula :- **sum = l1 + l2 + carry**

We did a lot of talk, let's understand it visually:-

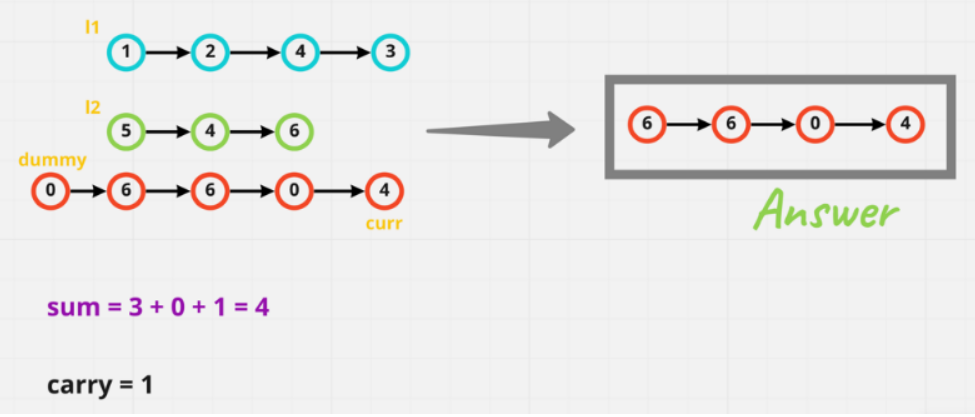
1st step->



2nd Step->



3rd Step->



Now I hope Ladies n Gentlemen, you got the crystal clear idea, what we are doing. So, without any further due ***let's code up***

class Solution {

public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

ListNode dummy = new ListNode(0); // creating an dummy list

ListNode curr = dummy; // intialising an pointer

int carry = 0; // intialising our carry with 0 intiall

// while loop will run, until l1 OR l2 not reaches null OR if they both reaches null. But our carry has some value in it.

// We will add that as well into our list

while(l1 != null || l2 != null || carry == 1){

int sum = 0; // intialising our sum

if(l1 != null){ // adding l1 to our sum & moving l1

sum += l1.val;

l1 = l1.next;

}

if(l2 != null){ // adding l2 to our sum & moving l2

sum += l2.val;

l2 = l2.next;

}

sum += carry; // if we have carry then add it into our sum

carry = sum/10; // if we get carry, then divide it by 10 to get the carry

ListNode node = new ListNode(sum % 10); // the value we'll get by moduloing it, will become as new node so. add it to our list

curr.next = node; // curr will point to that new node if we get

curr = curr.next; // update the current every time

}

return dummy.next; // return dummy.next bcz, we don't want the value we have consider in it intially!!

}

}

ANALYSIS :-

**Time Complexity :-** BigO(max(N, M)) where N is length of l1 & M is length of l2

**Space Complexity :-** BigO(max(N,M))