Add two LinkedList

- Hume ek essa function likhna hai, jisme Do linkedlist pass hongi, jo numbers ko represent kregi!
  phele element = most significant Digit
  last element = least Significant Digit
- Humare function Dono LinkedList ko add krengy aur new LinkedList return krengy!

MOST SIGNIFICANT ↘ LEAST SIGNIFICANT ↙

9 — 9 — 9 — 9
1 — 1
1 — 0 — 0 — 1 — 0

★ Hume Kya nahi krna ★

- Hume LinkedList ko reverse nahi krna!
- Hume array ya explicit extra money use nahi krna!
- Hume list ko number me convert krkey add nahi krna!

★ CONSTRAINTS ★

- Time Complexity : O(n)
- Space Complexity = Recursion space = O(n)

∴ 9⁴ — 7³ — 8² — 5¹
4² — 6¹

- Jab tak one or two ki place value same nahi hojati, tab tak hum unhe add nahi kregy!
- Hum, one ki lesser significant digit pe jayengy kuki one ki place value jayada!
- Dono same hone par bhi hum add nahi krengy kuki humey start krna hai least-significant se aur jana hai most significant tak.

one	two
4-0	4-0
5-1	6-1
8-2	4-2
7-3	4-2
9-4	4-2

Up the Stack (PUSH)

- Humne stack banaya, aur add krdia Dono list ke most significant digit unki place values ke sath.
- Hum Dono pointers ko null tak pochengy.

Down the Stack (POP)

- Jaisey hi Base condition hit hogi, hum carry (0) return krengy!
- ∴ sum = one.data + two.data + carry] Jab place value same hogi (pv1 = pv2)
- ∴ sum = one.data + carry] (pv1 > pv2)
- ∴ sum = two.data + carry] (pv2 > pv1)
- new carry ⇒ `c = sum / 10;`
- new digit ⇒ `d = sum % 10` ⇒ jo new linked list me add hogi (res → new linked list)
- res.addFirst
- carray ko return krdengy ke liye

pv1 = place value of 1
pv2 = place value of 2

BASE CASE			
4-0	4-0	0	
5-1	6-1	1	C
7-2	4-2	1	A
8-3	4-2	0	R
9-4	4-2	0	R
one	two		Y

CARRY 0 0 1 1 0
9 — 8 — 7 — 5
4 — 6

(9) → (9) → (2) → (1)

```
private static int addhelper(Node one, int pv1, Node two, int pv2,
                             LinkedList res)
{
  if(one==null && two=null){
        return 0;
  }
  int sum = 0;
  if(pv1 > pv2) {
  int oc = addhelper(one.next, pv1-1, two, pv2, res);
  sum = one.data + oc;
  }
  else if(pv2 > pv1) {
  int oc = addhelper(one, pv1, two.next, pv2-1, res);
  sum = two.data + oc;
  }
  else {                    oc = old carry
  int oc = addhelper(one.next, pv1-1, two.next, pv2-1, res);
  sum = one.data + two.data + oc;
  }
  int c = sum/10; // new carry
  int d = sum%10; // new digit for "res"
  res.addFirst(d);
  return c;
}

public static LinkedList addTwoLists(LinkedList one, LinkedList two){
  LinkedList res = new LinkedList();
  int oc = addhelper(one.head, one.size, two.head, two.size);
  if (oc>0) {
     res.addFirst(oc);
  }
  return res;
}
```