Add two numbers as Linked list

This is a standard question. Some edge cases to consider are what if both linked list are of unequal length and what if the linked list are of equal length but a carry is generated which make the answer longer than the length of linked list. To handle all these edge cases. We create a dummy node for the answer linked list that acts as a head pointer of the ans and a temp to traverse through the linkedlist. We check whether we have any operations to perform that is whether there exist a carry or a lis1 elem or list 2 elem if it does then we add it to sum. and move the list1 and list2 pointers respectively.

Carry is calculates by dividing sum by 10 and sum as sum%10. for sum we create a new node and temp→next points to this new node we then move the temp pointer to this new node waiting for whether or not more nodes need to be added.

When loop ends we have our ans and we just return the dummy→next.

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* list1, ListNode* list2) {
    ListNode* dummy=new ListNode(0);
    ListNode* temp=dummy;
    int sum=0,carry=0;
    while(list1!=NULL || list2!=NULL || carry)
    {
        sum=0;
        if(list1)
        {
            sum+=list1->val;
            list1=list1->next;
        }
        if(list2)
        {
            sum+=list2->val;
            list2=list2->next;
        }
```

```
        sum+=carry;
        carry=sum/10;
        ListNode* newNode=new ListNode(sum%10);
        temp->next=newNode;
        temp=temp->next;
    }
    return dummy->next;
    }
};
```

- Time Complexity : O(max(l1,l2)) whichever list is of maximum size the loop will run till that pointer.
- Space Complexity : O(1).