Class LLNode is used to store a node in a singly linked list, described on the following:

class LLNode {

public:

int val;

LLNode\* next;

LLNode();

LLNode(int val, LLNode\* next);

}

Where val is the value of node, next is the pointer to the next node.

**Request:** Implement function:

LLNode\* addLinkedList(LLNode\* l0, LLNode\* l1);

Where l0, l1 are two linked lists represented positive integers, each node is a digit, the head is the least significant digit (the value of each node is between 0 and 9, the length of each linked list is between 0 and 100000). This function returns the linked list representing the sum of the two integers.

**Example:**

Given l0 = [2, 3] (representing 32) and l1 = [1, 8] (representing 81). The result would be l0 = [3, 1, 1] (representing 32 + 81 = 113).

**Note:**

In this exercise, the libraries iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered\_map, set, unordered\_set, functional, algorithm has been included and namespace std are used. You can write helper functions and classes. Importing other libraries is allowed, but not encouraged, and may result in unexpected errors.

**For example:**

| **Test** | **Result** |
| --- | --- |
| int arr1[] = {2, 9};  int arr2[] = {0, 5};  LLNode\* head1 = LLNode::createWithIterators(arr1, arr1 + sizeof(arr1) / sizeof(int));  LLNode\* head2 = LLNode::createWithIterators(arr2, arr2 + sizeof(arr2) / sizeof(int));  LLNode\* newhead = addLinkedList(head1, head2);  LLNode::printList(newhead);  head1->clear();  head2->clear();  newhead->clear(); | [2, 4, 1] |