Add two numbers represented by linked lists | Set 1

Given two numbers represented by two lists, write a function that returns sum list. The sum list is list representation of addition of two input numbers.

Example 1

```
Input:
  First List: 5->6->3  // represents number 365
  Second List: 8->4->2 //  represents number 248
Output
  Resultant list: 3->1->6  // represents number 613
```

Example 2

```
Input:
  First List: 7->5->9->4->6  // represents number 64957
  Second List: 8->4 //  represents number 48
Output
  Resultant list: 5->0->0->5->6  // represents number 65005
```

Solution
Traverse both lists. One by one pick nodes of both lists and add the values. If sum is more than 10 then make carry as 1 and reduce sum. If one list has more elements than the other then consider remaining values of this list as 0. Following is C implementation of this approach.

```
#include<stdio.h>
#include<stdlib.h>

/* Linked list node */
struct node
{
    int data;
    struct node* next;
};

/* Function to create a new node with given data */
struct node *newNode(int data)
{
    struct node *new_node = (struct node *) malloc(sizeof(struct node));
    new_node->data = data;
    new_node->next = NULL;
    return new_node;
}

/* Function to insert a node at the beginning of the Doubly Linked List */
void push(struct node** head_ref, int new_data)
{
    /* allocate node */
    struct node* new_node = newNode(new_data);

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref)    = new_node;
}

/* Adds contents of two linked lists and return the head node of resultant li
struct node* addTwoLists (struct node* first, struct node* second)
{
    struct node* res = NULL; // res is head node of the resultant list
    struct node *temp, *prev = NULL;
    int carry = 0, sum;

    while (first != NULL || second != NULL) //while both lists exist
    {
        // Calculate value of next digit in resultant list.
        // The next digit is sum of following things
        // (i)  Carry
        // (ii) Next digit of first list (if there is a next digit)
        // (ii) Next digit of second list (if there is a next digit)
        sum = carry + (first? first->data: 0) + (second? second->data: 0);

        // update carry for next calulation
        carry = (sum >= 10)? 1 : 0;

        // update sum if it is greater than 10
        sum = sum % 10;

        // Create a new node with sum as data
        temp = newNode(sum);

        // if this is the first node then set it as head of the resultant lis
        if(res == NULL)
            res = temp;
        else // If this is not the first node then connect it to the rest.
            prev->next = temp;

        // Set prev for next insertion
        prev  = temp;

        // Move first and second pointers to next nodes
        if (first) first = first->next;
        if (second) second = second->next;
    }

    if (carry > 0)
      temp->next = newNode(carry);

    // return head of the resultant list
    return res;
}

// A utility function to print a linked list
void printList(struct node *node)
{
    while(node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
    printf("\n");
}

/* Drier program to test above function */
int main(void)
{
    struct node* res = NULL;
    struct node* first = NULL;
    struct node* second = NULL;
    // create first list 7->5->9->4->6
    push(&first, 6);
    push(&first, 4);
    push(&first, 9);
    push(&first, 5);
    push(&first, 7);
    printf("First List is ");
    printList(first);

    // create second list 8->4
    push(&second, 4);
    push(&second, 8);
    printf("Second List is ");
    printList(second);

    // Add the two lists and see result
    res = addTwoLists(first, second);
    printf("Resultant list is ");
    printList(res);

   return 0;
}
```

Output:

```
First List is 7 5 9 4 6
Second List is 8 4
Resultant list is 5 0 0 5 6
```

Time Complexity: O(m + n) where m and n are number of nodes in first and second lists respectively.

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

Related Topics:

- Clone a linked list with next and random pointer | Set 2
- Given a linked list of line segments, remove middle points
- Construct a Maximum Sum Linked List out of two Sorted Linked Lists having some Common nodes
- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links
- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List

Like 11 Tweet 0 +1 0

Writing code in comment? Please use **ideone.com** and share the link here.

71 Comments GeeksforGeeks Login

Recommend Share Sort by Newest

Join the discussion…

Ajitesh Mandal • 12 days ago

using dummy node

http://ideone.com/bdzXfe

Reply • Share ›

dipak • 13 days ago

#Simple solution using stacks and queues :(working code)

```
#include<iostream>
#include<queue>
#include <stdio.h>
#include <stdlib.h>
#include<algorithm>
#include<stack>
using namespace std;
// A linked List Node
struct node
{
int data;
struct node* next;
};

typedef struct node node;

/* A utility function to insert a node at the beginning of linked list */
```

see more

Reply • Share ›

surbhijain93 • 2 months ago

My code:

https://ideone.com/l5d2Yy

Reply • Share ›

meghasyam • 2 months ago

```
void addNumLink(Node *l1,Node *l2)
{
```

```
Node *t1=l1,*t2=l2,*l3=NULL;
int sum=0, i=1,car=0,count=1;
while ((t1!=NULL) && (t2!=NULL))
{
if (car==1)
{sum=(t1->x)+(t2->x)+1;car=0;}
else
sum=t1->x+t2->x;

if (sum>=10) {sum=sum%10;car=1; }
insertList(&l3,sum);

t1=t1->next;
t2=t2->next;
}

if (t1==NULL)
```

see more

• Reply • Share ›

laxxon • 2 months ago

If carry>0 remove that and it works fine if it is no then there arises prev->next and temp->next

• Reply • Share ›

Anurag Yadav • 2 months ago

@geeksforgeeks

while (first != NULL || second != NULL) //while both lists exist

comments should be //while any list exists
as the loop will continue to run even if one of the list gets NULL

1 • Reply • Share ›

Shivran Roy • 3 months ago

Another simple approach using dumy node

1. Get the number represented by link list 1 in num1

2.Get the number represented by link list 2 in num2

3.Add both num1 and num2 and represent the result in new link list using dummy node

void add_no(struct node *head1,struct node *head2)// the head of two link list to be added is provided

{

```
int num1,num2,result,n,i=1;

struct node *current,*temp;

struct node dumy;// dumy node used for adding nodes in resultant link list

struct node *tail=&dumy;
```

see more

1 • Reply • Share ›

PJT • 4 months ago

I just learned Link List. I tried the above code. It did work. But, there is one thing confused me. It looks like there is no connection between res and prev. Why res can be a head of resultant linked list? Can anyone explain this more? Thanks.

• Reply • Share ›

Anurag Singh PJT • 4 months ago

Please look at code carefully and you will see what's going on.
In while loop, at any time, we will take two nodes, sum them, create a new node with sum value and link this new node in resultant list. We link new sum node at the end of resultant list. For that we use prev, which always points to the last node in resultant list. res has nothing to do with prev. res points to very first node of the resultant list. And so at the end, we return res.

1 • Reply • Share ›

PJT Anurag Singh • 4 months ago

Anurag, thanks!

• Reply • Share ›

Tintin • 7 months ago

When we add 5->6->3 and 8->4->2, i thought the result is 13 -> 0 ->5

correct me if my understanding is wrong. The moment i saw the question, this came to my mind. But when i saw the discussed solution, i was surprised to learn that the addition started from right instead of left.

• Reply • Share ›

Siya Tintin • 6 months ago

Same came in my mind !!May be for the sake of simplicity they have done addition from right to left..

• Reply • Share ›

Pranav Kumar Jha • 7 months ago

A recursive solution :

http://ideone.com/U0wrKK

time complexity : O(max(m,n))

1 ^ | ⌄ • Reply • Share ›

Guest • 7 months ago

```
#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

struct node

{

int data;

struct node *next;

};

void addnode(int ch,struct node **head)

{

static struct node *tail=NULL;
```

see more

^ | ⌄ • Reply • Share ›

Parush Garg • 7 months ago

In carry we may write (sum/10) instead of 1.
It will work for adding numbers which are giving greater than 20 values.

^ | ⌄ • Reply • Share ›

Ishmeet Singh ➦ Parush Garg • 7 months ago

number can't be greater than 18. See for yourself, the max single digit is 9 and 9+9 is 18. So no need for that.

^ | ⌄ • Reply • Share ›

Parush Garg ➦ Ishmeet Singh • 7 months ago

I told for the General Inputs. They may insert 2 Digit values. :)

^ | ⌄ • Reply • Share ›

Abhishek Tamhane ↪ Parush Garg • 5 months ago

The Linked List represents a number in base 10. Each node is a digit. A digit can be anything from 0 to 9.

5->6->7 represent the number 765.

• Reply • Share ›

Yeshwanth Selvaraj • 8 months ago

http://ideone.com/9Cblsh

this will be the easiest method.Using recursion stack with complexity of O(m+n)

2 • Reply • Share ›

Kim Jong-il • 8 months ago

@GeeksforGeeks Time complexity will be O(max(m,n)).

• Reply • Share ›

AYUSH KUMAR • 8 months ago

this is my code

http://ideone.com/O5frCA

• Reply • Share ›

Manraj Singh • 9 months ago

Here is my code:http://ideone.com/D4s0yW

• Reply • Share ›

vipinkaushal • 9 months ago

why time complexity is O(m+n)
i think it's O(max(m,n))
please clear the doubt

thanks

2 • Reply • Share ›

skapp ↪ vipinkaushal • 7 months ago

time complexity is correct because we always check for the worst case analysis and worst case occurs when both lists are of same length,just give a thought you will get your doubt clear . Let me explain you if say list contains 4 nodes and list contains 7 nodes then you just have to care for the carry and sum till the 4 and 5th nodes and for 6th and 7th node there is no need for summing up data, whereas if both list would have been same nodes you have to keep a check on carry as well as summing up the data till the end of the both lists ,so i think its clear now why its O(m+n) .

• Reply • Share ›

Manraj Singh ➦ vipinkaushal • 9 months ago

same doubt

• Reply • Share ›

geekcoder08 • 9 months ago

Solution using Recusrion

http://ideone.com/e.js/SBfmBF

Time Complexity :O(n) where n is size of bigger Linked List
Space Complexity : O(n) ignoring stack frame memory

• Reply • Share ›

Gautam Goyal • 10 months ago

Recursive Approach:
typedef struct node* Link;
Link sumList(Link head1, Link head2){

int carry = 0;

int i=0;

int l1 = getLength(head1);

int l2 = getLength(head2);

int diff;

Link toInsert;

if(l1 > l2){

diff = l1-l2;

see more

2 • Reply • Share ›

ashish jaiswal ➦ Gautam Goyal • 3 months ago

Getting length of each list makes your code inefficient..

• Reply • Share ›

kinshuk chandra • a year ago

Here is the code from http://k2code.blogspot.in/2009...

```
node *long_add(mynode *h1, mynode *h2, mynode *h3)    //h3 = h2+h1
{
```

```
{
  node *c, *c1, *c2;
  int sum, carry, digit;

  carry = 0;
  c1 = h1->next;
  c2 = h2->next;

  while(c1 != h1 && c2 != h2)
  {
     sum   = c1->value + c2->value + carry;
     digit = sum % 10;
     carry = sum / 10;

     h3 = insertNode(digit, h3);
```

see more

• Reply • Share ›

Vishal Kumar Tiwari • a year ago

I have tried as well... the simplest logic :)

```
ode *AddTwoList(Node *head, Node *head2)
{
Node *i = head;
Node *j = head2;
int decimal = 1;
int sum1 =0;
int sum2 =0;
while(i || j)
{
if(i){
sum1 = sum1 +(i->info)*decimal;
i = i->next;
}
if(j){
sum2 = sum2 +(j->info)*decimal;
j = j->next;
```

see more

• Reply • Share ›

Vishal Kumar Tiwari • a year ago

this program will work only for single digit numbers......... ????

• Reply • Share ›

Reply • Share ›

ashish jaiswal ↪ Vishal Kumar Tiwari • 3 months ago

Yes..But you can make changes to this program to make it work for more than one digits also...

Reply • Share ›

Mohaan Raja • a year ago

Hi I have a solution. Please give your comments...
// Getting the actual number from the list for two lists
// Adding the numbers directly and getting the sum.
// Then creating the resultant list from the sum.

```
struct node
{
int data;
struct node *next ;
};

struct node * AddTwoList(struct node* a, struct node * b)
{
if(a==null)
return b;
if(b==null)
return a;

int num1 = GetNum(a);
```

see more

Reply • Share ›

Jayanth • a year ago

An even more simpler and neat solution

```
ListNode *addTwoNumbers(ListNode *l1, ListNode *l2) {

if(l1 == NULL && l2==NULL)

return NULL;

else if(l1 == NULL)

return l2;

else if(l2 == NULL)

return l1;
```

return add(l1,l2,0);

}

ListNode *add(ListNode *l1,ListNode *l2,int carry)

see more

1 • Reply • Share ›

yeefoo • a year ago

Pretty neat, thanks!
One comment, you can create a dummy node for result to avoid unnecessary branches. The code looks more cleaner that way in my opinion.

• Reply • Share ›

Guest • a year ago

On a similar note, how do you solve this when your LL pointer points to the MSB, and you are asked to subtract two large linked lists without reversing them?

• Reply • Share ›

Reymark Dahao • 2 years ago

tig nan niu tenga ni ate shara ang laki/

1 • Reply • Share ›

TOm Garcia • 2 years ago

for real?

• Reply • Share ›

Rajdeep • 3 years ago

- One easy solution is like the one being used to print Link List in reverse direction and create the numbers from the two list
- Then add the two lists number.

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

Rajdeep Rajdeep • 3 years ago

Only printing the sum and not creating the sum list:

Please find the code for the above algo:

```
long FindNumInList(struct node *head)
{
   long num=0;

   if(!head)
         return;

   while(head)
   {
      num = 10*num + head->data;
      head = head->next;
   }
```

see more

^ | v • Reply • Share ›

Amateur_coder ➦ Rajdeep • 2 years ago

simple and smart....

```
/* Paste your code here (You may delete these lines if not writing code) */
```

^ | v • Reply • Share ›

nikoo28 • 3 years ago

can there be an efficient way to multiply to numbers represented by linked lists..?? i mean like 3->6->4->7->3 (multiplied by) 5->2->7->1->1->7

if anyone can..please help me..as the carry can be anything here except 0 or 1..also we need to add up results..please help..

^ | v • Reply • Share ›

Ankit Malhotra ➦ nikoo28 • 2 years ago

You can make a function to multiply a linked list with a single digit extracted from the other list. You can add 0 to the result according to the place value of the digit in the list. You then need to add all the results using the addition solution given in this page.

^ | v • Reply • Share ›

bhavneet • 3 years ago

using recursion

```
 struct node* add(struct node* list1, struct node* list2)
{
        int sum=0;
        static int carry=0;
        struct node* temp;
        if(list1==NULL && list2==NULL && carry==0)
                return NULL;
        if(list1!=NULL)
                {
                        sum+=list1->info;
                        list1=list1->next;


                }
         if(list2!=NULL)
                {
                        sum+=list2->info;
```

see more

• Reply • Share ›

fender bender • 3 years ago

new_node, free_list and print_list are generic fns, and i haven't shown them here for conciseness.

```
 void addition_list (node_t *lista, node_t *listb)
{
    node_t *result = NULL;
    node_t *new = NULL;
    int carry = 0;
    int sum = 0;

    while (lista != NULL || listb != NULL) {
        sum = 0;

        if (lista != NULL) {
            sum = lista->data;
            lista = lista->next;
        }
```

see more

• Reply • Share ›

ANSHUM AGARWAL • 3 years ago

#include
#include
using namespace std;
typedef list l;
int main()
{

```
{
l l1;
l l2;
l l3;
l::iterator it1;
l::iterator it2;
l::iterator it3;
int n,m,i,j,t,k,l;
cout<>n;
cout<<"\nenter the no:\n";
for(i=0;i>t;
l1.push_front(t);
}
```

see more

^ | ⌄ • Reply • Share ›

ANSHUM AGARWAL • 3 years ago

```
#include
#include
using namespace std;
typedef list l;
int main()
{
l l1;
l l2;
l l3;
l::iterator it1;
l::iterator it2;
l::iterator it3;
int n,m,i,j,t,k,l;
cout<>n;
cout<<"\nenter the no:\n";
for(i=0;i>t;
l1.push_front(t);
}
```

see more

^ | ⌄ • Reply • Share ›

Venkata Krishna • 3 years ago

```
int sum (struct node *head)
{
int count = 0, sum = 0;
struct node *temp = head;
```

while(teamp!=NULL)
{
sum = sum + (temp->data * 10 ^ count);

temp = temp->next;

count++;
}
return sum;
}

use above function for two lists and both return sums. I hope it will solve the problem well.

Any one let me know the complexity of the above program.

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

anuj • 3 years ago

```
 #include<stdio.h>
#include<stdlib.h>

struct node {
        int data;
        struct node * next;
};

void push(struct node ** href, int data) {
        struct node * newnode = (struct node *) malloc (sizeof(struct node));
        newnode -> data = data;
        newnode -> next = *href;
        *href = newnode;
}

void printList(struct node * head) {
        while(head != NULL){
                printf("%d ", head->data);
```

see more

• Reply • Share ›

amitp49 • 3 years ago

For order linked list, we can add them without reversing them too using recursive approach by maintaining reverse carry(rc) which will be return back from previous calls...

Here is the function for running code..

Please let me know if any case it won't work...

```
 /* Paste your code here (You may delete these lines if not writing code) */



struct node* addTwoLists_order_rec(struct node* first, struct node* second,int fc,int sc,in
{
        if(first == NULL && second == NULL)
        {
                *rc = 0;
                return NULL;
        }

        struct node *res = newNode(0);
```

see more

Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

GeeksforGeeks

Like

93,009 people like GeeksforGeeks.

- Facebook social plugin
-
-
    - Interview Experiences
    - Advanced Data Structures
    - Dynamic Programming
    - Greedy Algorithms
    - Backtracking
    - Pattern Searching
    - Divide & Conquer
    - Mathematical Algorithms
    - Recursion
    - Geometric Algorithms
-

- ## Popular Posts

    - All permutations of a given string
    - Memory Layout of C Programs
    - Understanding “extern” keyword in C
    - Median of two sorted arrays
    - Tree traversal without recursion and without stack!
    - Structure Member Alignment, Padding and Data Packing
    - Intersection point of two Linked Lists
    - Lowest Common Ancestor in a BST.
    - Check if a binary tree is BST or not
    - Sorted Linked List to Balanced BST

- Follow @GeeksforGeeks Subscribe

- ## Recent Comments

    - acemrek

      We can also do it with keeping parent....

      Given a binary tree, how do you remove all the half nodes? · 1 hour ago

    - acemrek

      R1 Q1: http://ideone.com/EUHNGh

Snapdeal Interview Experience | Set 12 (For Senior Software Developer) · 2 hours ago

- Mrinmay Mukherjee

  Hope it helps :) Please share your suggestions!...

  C++ Programming Language · 2 hours ago

- Praveen Dara

  //Author Praveen Dara (AITP) //counts the...

  Write a function that counts the number of times a given int occurs in a Linked List · 3 hours ago

- Praveen Dara

  //Author Praveen Dara (AITP) // deletion of SLL...

  Write a function to delete a Linked List · 3 hours ago

- Aditya Goel

  Method#4 Traverse array once and find out min,...

  Check if array elements are consecutive | Added Method 3 · 3 hours ago