Add two numbers represented by linked lists | Set 2

Given two numbers represented by two linked lists, write a function that returns sum list. The sum list is linked list representation of addition of two input numbers. It is not allowed to modify the lists. Also, not allowed to use explicit extra space (Hint: Use Recursion).

Example

```
Input:
  First List: 5->6->3  // represents number 563
  Second List: 8->4->2 //  represents number 842
Output
  Resultant list: 1->4->0->5  // represents number 1405
```

We have discussed a solution here which is for linked lists where least significant digit is first node of

lists and most significant digit is last node. In this problem, most significant node is first node and least significant digit is last node and we are not allowed to modify the lists. Recursion is used here to calculate sum from right to left.

Following are the steps.
1) Calculate sizes of given two linked lists.
2) If sizes are same, then calculate sum using recursion. Hold all nodes in recursion call stack till the rightmost node, calculate sum of rightmost nodes and forward carry to left side.
3) If size is not same, then follow below steps:
….**a)** Calculate difference of sizes of two linked lists. Let the difference be *diff*
….**b)** Move *diff* nodes ahead in the bigger linked list. Now use step 2 to calculate sum of smaller list and right sub-list (of same size) of larger list. Also, store the carry of this sum.
….**c)** Calculate sum of the carry (calculated in previous step) with the remaining left sub-list of larger list. Nodes of this sum are added at the beginning of sum list obtained previous step.

Following is C implementation of the above approach.

```
// A recursive program to add two linked lists

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

// 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 */
void push(struct node** head_ref, int new_data)
{
    /* allocate node */
    struct node* new_node = (struct node*) malloc(sizeof(struct node));

    /* put in the data  */
    new_node->data  = 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;
}

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

```
        node = node->next;
    }
    printf("\n");
}

// A utility function to swap two pointers
void swapPointer( node** a, node** b )
{
    node* t = *a;
    *a = *b;
    *b = t;
}

/* A utility function to get size of linked list */
int getSize(struct node *node)
{
    int size = 0;
    while (node != NULL)
    {
        node = node->next;
        size++;
    }
    return size;
}

// Adds two linked lists of same size represented by head1 and head2 and retu
// head of the resultant linked list. Carry is propagated while returning fro
// the recursion
node* addSameSize(node* head1, node* head2, int* carry)
{
    // Since the function assumes linked lists are of same size,
    // check any of the two head pointers
    if (head1 == NULL)
        return NULL;

    int sum;

    // Allocate memory for sum node of current two nodes
    node* result = (node *)malloc(sizeof(node));

    // Recursively add remaining nodes and get the carry
    result->next = addSameSize(head1->next, head2->next, carry);

    // add digits of current nodes and propagated carry
    sum = head1->data + head2->data + *carry;
    *carry = sum / 10;
    sum = sum % 10;

    // Assigne the sum to current node of resultant list
    result->data = sum;

    return result;
}
```

```
// This function is called after the smaller list is added to the bigger
// lists's sublist of same size.  Once the right sublist is added, the carry
// must be added toe left side of larger list to get the final result.
void addCarryToRemaining(node* head1, node* cur, int* carry, node** result)
{
    int sum;

    // If diff. number of nodes are not traversed, add carry
    if (head1 != cur)
    {
        addCarryToRemaining(head1->next, cur, carry, result);

        sum = head1->data + *carry;
        *carry = sum/10;
        sum %= 10;

        // add this node to the front of the result
        push(result, sum);
    }
}

// The main function that adds two linked lists represented by head1 and head
// The sum of two lists is stored in a list referred by result
void addList(node* head1, node* head2, node** result)
{
    node *cur;

    // first list is empty
    if (head1 == NULL)
    {
        *result = head2;
        return;
    }

    // second list is empty
    else if (head2 == NULL)
    {
        *result = head1;
        return;
    }

    int size1 = getSize(head1);
    int size2 = getSize(head2) ;

    int carry = 0;

    // Add same size lists
    if (size1 == size2)
        *result = addSameSize(head1, head2, &carry);

    else
    {
```

```
        int diff = abs(size1 - size2);

        // First list should always be larger than second list.
        // If not, swap pointers
        if (size1 < size2)
            swapPointer(&head1, &head2);

        // move diff. number of nodes in first list
        for (cur = head1; diff--; cur = cur->next);

        // get addition of same size lists
        *result = addSameSize(cur, head2, &carry);

        // get addition of remaining first list and carry
        addCarryToRemaining(head1, cur, &carry, result);
    }

    // if some carry is still there, add a new node to the fron of
    // the result list. e.g. 999 and 87
    if (carry)
        push(result, carry);
}

// Driver program to test above functions
int main()
{
    node *head1 = NULL, *head2 = NULL, *result = NULL;

    int arr1[] = {9, 9, 9};
    int arr2[] = {1, 8};

    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    // Create first list as 9->9->9
    int i;
    for (i = size1-1; i >= 0; --i)
        push(&head1, arr1[i]);

    // Create second list as 1->8
    for (i = size2-1; i >= 0; --i)
        push(&head2, arr2[i]);

    addList(head1, head2, &result);

    printList(result);

    return 0;
}
```

Output:

```
1  0  1  7
```

Time Complexity: O(m+n) where m and n are the sizes of given two linked lists.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

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

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

May Oo • 4 days ago

can you pls guide to the correct response for the following question?

The
below function `alt_sum` was written to return the sum of ALTERNATE
elements of a linked list, starting with the FIRST element.

typedef struct node {

int data;

struct node *next;

} node_t;

int alt_sum(node_t *head) {

int sum = 0;

```
while(head != NULL) {
```

see more

• Reply • Share ›

Chakra Agarwal • 21 days ago

Another way of doing this will be reverse both the linked list and start to add them. Each node created after adding will be added as head in new linked list.
Eg:
List1: 1->2->3->4
List 2: 1->2
reverse list1: 4->3->2->1
reverse list 2: 2->1
First node 6 is created
then 4->6
and so on
1->2->4->6

• Reply • Share ›

Nishith Goswami • 2 months ago

If in case size of link list is not same in that case..can't we update new nodes with data value zero and follow the same recursive approach.

For Ex:
L1 : 1->2->3->4
L2 : 2->3->4

Changed L2 like : 0->2->3->4

and follow same recursive approach.

1 • Reply • Share ›

Vj • 3 months ago

Why do you need to pass cur in addCarryToRemaining(head1, cur, &carry, result) ?

• Reply • Share ›

sneha ➚ Vj • a month ago

To stop the recursion. In addCarryToRemaining(), we are starting from head1 and traversing head1->next till head1 gets equal to cur.

• Reply • Share ›

Ashish Jaiswal • 3 months ago

simple code ...No recursion...iterative...

complexity:O(m+n)
sapce: O(1)

```
#include<stdio.h>
#include<stdlib.h>
struct node;
void push(struct node**,int);
void print(struct node*);
struct node*add(struct node*,struct node*);
struct node*newnode();

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

int main()
```

see more

1 • Reply • Share ›

Ashish Jaiswal Ashish Jaiswal • 3 months ago

But it will have problem when given no is too big....so better take double or long double the variable in which we are storing the no...

• Reply • Share ›

Harit • 4 months ago

```
package com.learner.questions.others;

public class LinkedListSum {

public static void main(final String[] args) {

final Node numberOne = buildList(new int[] { 2, 4 });

final Node numberTwo = buildList(new int[] { 1, 8, 9 });

final Node sum = getSum(numberOne, numberTwo);

print(numberOne);

print(numberTwo);

print(sum);

}
```

private static int getLength(Node node) {

see more

• Reply • Share ›

Ashish • 5 months ago

Another approach can be traverse the lists and get the integers. Now add these and make a new Linked List. Efficient with respect to time as well as space.

• Reply • Share ›

Guest ➚ Ashish • 5 months ago

What will you do when the integers are really very large that can't be stored in the variable.

3 • Reply • Share ›

Ashish Jaiswal ➚ Guest • 3 months ago

We can take Double or long Double in case no is big...

• Reply • Share ›

zxcvmnb • 5 months ago

If recursion is possible solution, using explicit array(stack) or better still converting lists to ints should also be solution. :)

• Reply • Share ›

jimmy • 6 months ago

what if lists are..1>2>3 AND 9>9>9....then ans should be 1>2>2>2 but above code will give 1>2>2..please check it out

• Reply • Share ›

Nayan • 6 months ago

Mine is simple implementation

Link of code : http://ideone.com/IB9YbA

• Reply • Share ›

Kim Jong-il ➚ Nayan • 6 months ago

Is it? :D :D I doubt.:P

• Reply • Share ›

Rajat Shrivastava • 7 months ago

Java implementation of above algorithm

http://ideone.com/7T0LJU

• Reply • Share ›

kamran siddique • 7 months ago

success

http://ideone.com/ZHs8kH

• Reply • Share ›

Ashish Jaiswal kamran siddique • 3 months ago

Order is not maintained...you just printing reverse of linked at last while no should be of same order as sum is..

```
void rprint(struct node*head){
if(head==NULL)
return ;
rprint(head->next);
printf("%d ",head->data);
}
```

• Reply • Share ›

Guest • 8 months ago

Reverse both , add them starting from head to tail. then reverse the resulting list???? Is it correct

1 • Reply • Share ›

pol Guest • 8 months ago

condition is u can't modify the original list.so reversing is not possible.

• Reply • Share ›

sk pol • 6 months ago

reverse all again.

• Reply • Share ›

Pankaj sk • 3 months ago

When we say we can't modify the DS, we imply it can be a read-only DS.

• Reply • Share ›

crankyCoder Guest • 8 months ago

would be correct...yes

• Reply • Share ›

faiz • 8 months ago

How about traversing the lists and storing them in two stacks. Keep popping from both until one of them becomes empty .. The addition logic is same

• Reply • Share ›

Ashish Jaiswal ➦ faiz • 3 months ago

yes....but extra space for both lists in stack

• Reply • Share ›

<HoldOnLife!#> • 8 months ago

I found it analogous to http://www.geeksforgeeks.org/s... is it?

• Reply • Share ›

prakhs123 • 9 months ago

http://ideone.com/KIUNTQ

Have a look at my tested code!

1 • Reply • Share ›

KeshaShah • 9 months ago

We can reverse 1st list, reverse 2nd list . Find solution as per previous question and again reverse the ans .. That better according to me..

12 • Reply • Share ›

debashis_deb ➦ KeshaShah • a month ago

i did the same approach few days back in an online hiring test :) !

• Reply • Share ›

Pankaj ➦ KeshaShah • 3 months ago

What if it's a Read-Only DS, or the memory location is shared across multiple processes?

• Reply • Share ›

badal rooprai • 9 months ago

http://ideone.com/38hvqB

here is my tested code and a short code **@GeeksforGeeks**

main function is like this

int add(struct node*a,struct node*b,struct node**c,int n,int m)

{if(n==0&&m==0)

return 0;

int sum;

if(n<m) {sum="add(a,b-">next,c,n,m-1);

sum+=b->data;

```
}

else if(n>m)
```

see more

• Reply • Share ›

danny • 9 months ago

We can reverse the list and then apply previous easy method, and then in end reverse the list again....

1 • Reply • Share ›

Ashish Jaiswal ➦ danny • 3 months ago

We can not modify list as given in problem...Its read only....

• Reply • Share ›

Harsh Agarwal • 9 months ago

Alternate approach.
Step1: Reverse both the lists.
Step 2: Now the problem has been reduced to a problem earlier solved.
Step 3: Reverse the resultant list to obtain the answer.

1 • Reply • Share ›

Ashish Jaiswal ➦ Harsh Agarwal • 3 months ago

We can not modify list as given in problem..

• Reply • Share ›

GOPI GOPINATH • a year ago

will the above approach work for addition of negative numbers too ??

• Reply • Share ›

RK ➦ GOPI GOPINATH • 8 months ago

If both are negative, yes.

• Reply • Share ›

gkns • a year ago

@GeeksforGeeks
A more slick way (I guess) to do it would be a single function:

```
static int carry; //function argument in case of pass by reference;
static LinkedNode<integer> prev = null; //argument in case of "
public static LinkedNode<integer> llAdd(LinkedNode<integer> ll1, LinkedNode<integer> ll2, i
{
```

```
{
        int a, b, sum;
        if (ll1 != null){
                a = ll1.item;
                ll1 = ll1.next;
        }
        else
                a = 0;
        if (ll2 != null && shift <= 0)
        {
                b = ll2.item;
```

see more

• Reply • Share ›

Aditya Gaurav • a year ago

s1=reverse(s1);s2=reverse(s2);
s* s3=addLists(s1,s2);s3=reverse(s3);
s3 is the required list...
function definition
s* addLists(s* s1, s* s2, int carry=0){

int sum=0;

if(s1!=NULL)sum+=s1->data;if(s2!=NULL)sum+=s2->data;sum+=carry;

if(sum==0)return NULL;

carry=sum/10;sum=sum%10;

s* temp=getnew(sum);

temp->link=addLists((s1==NULL)?NULL:s1->link,(s2==NULL)?s2:s2->link,carry);

return temp;

}

1 • Reply • Share ›

Aditya Gaurav • a year ago

// s1 is 1st list, s2 is 2nd list; s3 is the resultant list
Calling:
s1=rev(s1);s2=rev(s2); //rev() reverses the list
s* s3=addLists(s1,s2); // recursive function to get s3
s3=rev(s3);
display(s3); //display() displays the list node data

```
//recursive function definition;
s* addLists(s* s1, s* s2, int carry=0){
int sum=0;
if(s1!=NULL)sum+=s1->data;if(s2!=NULL)sum+=s2->data;sum+=carry;
if(sum==0)return NULL;
carry=sum/10;sum=sum%10;
s* temp=getnew(sum); //getnew(n) is utility function to get a new node with values as: data=n
link=NULL
temp->link=addLists((s1==NULL)?NULL:s1->link,(s2==NULL)?s2:s2->link,carry);
return temp;
}
```

• Reply • Share ›

Ashish Jaiswal ➚ Aditya Gaurav • 3 months ago

What if the given lists are Read only...??

• Reply • Share ›

Aadithya Umashanker ➚ Aditya Gaurav • 10 months ago

Isn't this method simple??

1 • Reply • Share ›

theCuriosityEnthusiast ➚ Aadithya Umashanker • 10 months ago

It says you can't modify the list right at the beginning :P

2 • Reply • Share ›

Aditya Gaurav ➚ theCuriosityEnthusiast • 7 months ago

then just reverse the s1 ans s2 lists once again and the original lists will be restored.. :P

• Reply • Share ›

Sambhav Sharma • a year ago

I think this is a pretty easy way to do it. What could be the possible drawbacks using this code?

```
void add(struct node *start1, struct node *start2)
{
struct node *tmp1=start1->next, *tmp2=start2->next;

int num1=start1->data,num2=start2->data,num3=start3->data;

int sum=0;

while(tmp1)
{
num1 = num1 * 10 + tmp1->data;
```

```
tmp1=tmp1->next;
}

while(tmp2)
{
```

see more

3 • Reply • Share ›

theCuriosityEnthusiast ➦ Sambhav Sharma • 10 months ago

If the number is incredibly huge and even long long integers can't handle the numbers, then this approach won't work..

1 • Reply • Share ›

Ashish Jaiswal ➦ theCuriosityEnthusiast • 3 months ago

Right....agreed..

• Reply • Share ›

faisal • a year ago

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node *next;

};

struct node *res=NULL;

/* Function to insert a node at the beginning */

void push(struct node ** head_ref, int new_data)

{
```

see more

• Reply • Share ›

Sameer • a year ago

http://codingrecipies.blogspot...,

Liked this blog , full code in java , simple to understand

Liked this blog , full code in java , simple to understand

• Reply • Share ›

Vishal Hemnani Sameer • a year ago

That is a hack to solve this question. I'd recommend using gforg's approach..

• Reply • Share ›

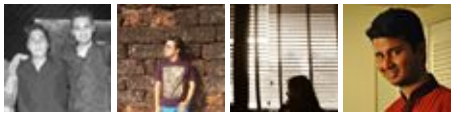
Load more comments

Subscribe Add Disqus to your site Privacy

- GeeksforGeeks

  Like

  93,017 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? · 5 hours ago

  - acemrek

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

    Snapdeal Interview Experience | Set 12 (For Senior Software Developer) · 5 hours ago

  - Mrinmay Mukherjee

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

    C++ Programming Language · 5 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 · 6 hours ago

  - Praveen Dara

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

    Write a function to delete a Linked List · 6 hours ago

  - Aditya Goel

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

Check if array elements are consecutive | Added Method 3 · 6 hours ago