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 list */
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 list
        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;
}
```

Run on IDE

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.

106 Comments Category: Linked Lists

Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Merge Sort for Doubly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data

(Login to Rate and Mark)

1.6 Average Difficulty : **1.6/5.0**
Based on **5** vote(s)

Add to TODO List
Mark as DONE

Like Share 13 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.