**Name**: Ravi Jamanbhai Makwana

**Roll No**: 21BCP418

Div-6, G-12

**Assignment-3**

**Aim:**

To implement addition of two numbers using linked list.

**Program:**

#include <stdio.h>

#include <stdlib.h>

struct node{

int info;

struct node\* next;

};

//function to insert number to a linked list

//its return type is a pointer object of the structure

struct node\* insert\_at\_end(int item, struct node \*head){

struct node \*temp; //created for traversal

struct node\* ptr = (struct node\*)malloc(sizeof(struct node)); //new node to be added

ptr->info = item;

if(head == NULL){

ptr -> next = NULL;

head = ptr;

}

else{

temp = head;

while (temp -> next != NULL){

temp = temp -> next;

}

temp->next = ptr;

ptr->next = NULL;

}

return head;

}

//function to add 2 linked lists and return the result as a 3rd linked list

struct node\* adder(struct node\* head1, struct node\* head2, struct node\* head3){

struct node\* temp1 = head1; //created for traversal of 1st linked list

struct node\* temp2 = head2; //created for traversal of 2nd linked list

int carry = 0; //initially carry = 0

while (temp1 != NULL){

head3 = insert\_at\_end(((temp1->info) + (temp2->info) + carry) % 10, head3);

if ((temp1->info + temp2->info + carry) > 9 ){

carry = 1;

}

else {

carry = 0;

}

//increment both linked lists

temp1 = temp1->next;

temp2 = temp2->next;

}

//if for MSB of both numbers bear a number greater than 9 on addition, then we need to add an extra 1 in the result

if (carry == 1){

head3 = insert\_at\_end(1,head3);

}

return head3;

}

void display (struct node\* head){

struct node\* temp = (struct node\*)malloc(sizeof(struct node));

temp = head;

while (temp != NULL){

printf("%d ",temp->info);

temp = temp->next;

}

printf("\n");

}

void display\_reverse(struct node\* head){

struct node\* temp = (struct node\*)malloc(sizeof(struct node));

temp = head;

if (temp != NULL){

display\_reverse(temp->next);

printf("%d ", temp->info);

}

}

int main(){

long n1,n2;

int size1=0,size2=0;

printf("Enter 1st value: ");

scanf("%d",&n1);

printf("Enter 2nd value: ");

scanf("%d",&n2);

//creating 3 linked-lists

struct node\* head1 = (struct node\*)malloc(sizeof(struct node)); //for n1

struct node\* head2 = (struct node\*)malloc(sizeof(struct node)); //for n2

struct node\* head3 = (struct node\*)malloc(sizeof(struct node)); //for result

head1 = NULL;

head2 = NULL;

head3 = NULL;

//inserting digits one by one into 1st linked-list

long temp\_digit = n1;

while (temp\_digit > 0){

head1 = insert\_at\_end(temp\_digit%10, head1);

temp\_digit /= 10;

size1++;

}

//inserting digits one by one into 2nd linked-list

temp\_digit = n2;

while (temp\_digit > 0){

head2 = insert\_at\_end(temp\_digit%10, head2);

temp\_digit /= 10;

size2++;

}

//inserting extra 0s at start to whichever number is smaller

if (size1>size2){

for (int i=0;i<(size1-size2);i++){

head2 = insert\_at\_end(0, head2);

}

}

else if (size2>size1){

for (int i=0;i<(size2-size1);i++){

head1 = insert\_at\_end(0, head1);

}

}

head3 = adder(head1,head2,head3);

display\_reverse(head1);

printf("\n");

display\_reverse(head2);

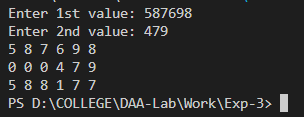
printf("\n");

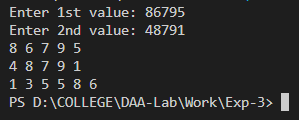
display\_reverse(head3);

return 0;

}

**Output:**





**Analysis:**

**Note**: Steps taking constant time are not considered.

Let us assume the length of both numbers entered is n (i.e. n digits)

**Time Complexity for insert\_at\_end method:**

The while loop runs for n times:

Time complexity: O(n)

**Time Complexity for adder method:**

While loop runs for n times.

For each iteration of while loop, method insert\_at\_end method is called for the new linked list.

So,

Time complexity: (1) + (2) + (3) + … + (n-1) + (n)

= n\*(n+1) / 2

= O(n2)

**Time Complexity for display method:**

The while loop runs for n times:

Time complexity: O(n)

**Time Complexity for display\_reverse method:**

Method is called recursively till all digits are traversed (i.e. for n times)

Time complexity: O(n)

**Time Complexity for main method: (total time complexity of the program):**

For insertion of digits of 1st number in linked list-1:

while loop runs for n times:

each time method insert\_at\_end is called:

Time complexity: (1) + (2) + (3) + … + (n-1) + (n)

= n\*(n+1) / 2

= O(n2)

For insertion of digits of 2nd number in linked list-2:

while loop runs for n times:

each time method insert\_at\_end is called:

Time complexity: (1) + (2) + (3) + … + (n-1) + (n)

= n\*(n+1) / 2

= O(n2)

Then, the method adder is called, which takes time as O(n2)

Then, the method display\_reverse is called 3 times with each time taking the time of O(n)

**Total Time Complexity**: O(n2) + O(n2) + O(n2) + 3\*O(n)

= 3\*O(n2) + 3\*O(n)

= O(n2)