Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of AI-ML

Experiment	3
Aim	Implement the given problem statement
Objective	Addition of numbers Given as input two numbers A,B as singly linked lists, output A + B as a singly linked list. Avoid the use of any data structures other than singly linked lists. Sample Input: 2 -> 8 -> 9 -> 9, 6 -> 1 -> 0 -> 1 Output: 9 -> 0 -> 0 -> 0
Name	Balla Mahadev Shrikrishna
UCID	2023300010
Class	A
Batch	A
Date of Submission	05-09-24

Explanation of the technique used	The technique used in this code involves storing digits in a linked list in the correct order. For ease of addition, the linked lists are reversed, allowing the addition to start from the units place. This approach simplifies the handling of carry-over during addition. After calculating the sum, the resultant linked list is reversed again to restore the digits to their original order.
	
	Flow of the program(code):
	1) Create linked lists from i/p arrays
	2) Reverse the linked lists
	3) Add them
	4) Reverse the returned result linked list
	5) Print and free the memory
	
	Algorithm for addition of linked lists:
	temp1 = n1, temp2 = n2, result = NULL, sum = 0, carry = 0
	while(temp1 || temp2 || carry){ //till the LLs aren't fully traversed and carry isn't present the following block is executed
	sum = carry
	if(temp1){
	sum += val(temp1)
	temp1 = next(temp1)
	}
	if(temp2){
	sum += val(temp2)
	temp2 = next(temp2)

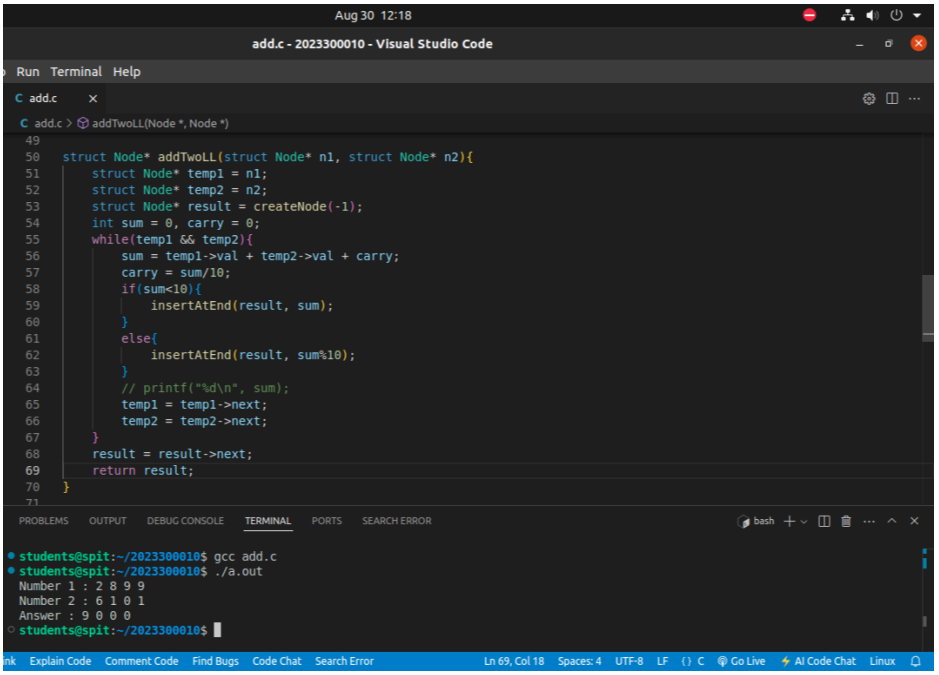
	} carry = sum/10 result = insertAtEnd(result, sum%10) }

Program(Code)	#include<stdio.h>
	#include<stdlib.h>
	
	struct Node{
	int val;
	struct Node* next;
	};
	
	struct Node* createNode(int val){
	struct Node* new = (struct Node*)malloc(sizeof(struct Node));
	new->val = val;
	new->next = NULL;
	return new;
	}
	
	struct Node* reverseLL(struct Node* head){
	struct Node* curr = head;
	struct Node* prev = NULL;
	struct Node* nex;
	while(curr != NULL){
	nex = curr->next;
	curr->next = prev;
	prev = curr;
	curr = nex;
	}
	return prev;
	}
	
	struct Node* insertAtEnd(struct Node* head,int val){
	struct Node* newNode = createNode(val);
	if(head == NULL){
	return newNode;
	}
	else{
	struct Node* temp = head;
	while(temp->next != NULL){
	temp = temp->next;
	}
	temp->next = newNode;
	return head;
	}
	}
	
	void printList(struct Node* node){
	while(node->next != NULL){
	printf("%d->", node->val);

```
        node = node->next;
    }
    printf("%d", node->val);
}

struct Node* addTwoLL(struct Node* n1, struct Node* n2){
    struct Node* temp1 = n1;
    struct Node* temp2 = n2;
    struct Node* result = NULL;
    int sum = 0, carry = 0;
    while(temp1 || temp2 || carry){
        sum = carry;
        if(temp1){
            sum += temp1->val;
            temp1 = temp1->next;
        }
        if(temp2){
            sum += temp2->val;
            temp2 = temp2->next;
        }
        // printf("%d\n", sum);
        carry = sum/10;
        result = insertAtEnd(result, sum%10);
    }
    return result;
}

struct Node* createLinkedListFromIntArray(int* arr, int size){
    struct Node* head = NULL;
    for(int i=0; i<=size-1; i++){
        head = insertAtEnd(head, arr[i]);
    }
    return head;
}

void freeLL(struct Node* head){
    struct Node* temp;
    while(head != NULL){
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main(){
    int num1[100], num2[100];
    int n1_size, n2_size;

    printf("Enter the number of digits in the first number : ");
    scanf("%d", &n1_size);
```

```
    printf("Enter the first number digit by digit : ");
    for(int i=0; i<n1_size; i++){
        scanf("%d", &num1[i]);
    }

    printf("Enter the number of digits in the second number : ");
    scanf("%d", &n2_size);
    printf("Enter the second number digit by digit : ");
    for(int i=0; i<n2_size; i++){
        scanf("%d", &num2[i]);
    }

    struct Node* n1 = createLinkedListFromIntArray(num1, n1_size);
    struct Node* m1 = createLinkedListFromIntArray(num2, n2_size);

    printf("Number 1 : ");
    printList(n1);
    printf("\n");
    printf("Number 2 : ");
    printList(m1);
    printf("\n");

    struct Node *result = addTwoLL(reverseLL(n1), reverseLL(m1));
    printf("Answer : ");
    printList(reverseLL(result));
    printf("\n");

    freeLL(n1); freeLL(m1); freeLL(result);
    return 0;
}
```

Output



```
C:\Mahadev\S.E\DS\Lab Sessions>gcc add.c

C:\Mahadev\S.E\DS\Lab Sessions>a.exe
Enter the number of digits in the first number : 4
Enter the first number digit by digit : 6 1 0 1
Enter the number of digits in the second number : 4
Enter the second number digit by digit : 2 8 9 9
Number 1 : 6->1->0->1
Number 2 : 2->8->9->9
Answer : 9->0->0->0

C:\Mahadev\S.E\DS\Lab Sessions>a.exe
Enter the number of digits in the first number : 7
Enter the first number digit by digit : 4 8 9 9 8 7 9
Enter the number of digits in the second number : 4
Enter the second number digit by digit : 9 9 9 9
Number 1 : 4->8->9->9->8->7->9
Number 2 : 9->9->9->9
Answer : 4->9->0->9->8->7->8

C:\Mahadev\S.E\DS\Lab Sessions>a.exe
Enter the number of digits in the first number : 4
Enter the first number digit by digit : 6 5 9 9
Enter the number of digits in the second number : 4
Enter the second number digit by digit : 7 8 3 4
Number 1 : 6->5->9->9
Number 2 : 7->8->3->4
Answer : 1->4->4->3->3

C:\Mahadev\S.E\DS\Lab Sessions>
```

Conclusion	The above program shows the practical application of linked lists for performing arithmetic operations on large numbers, which are stored as individual digits within the nodes.