# AVANISH RAJ SRIVASTAVA BT22CSH031 ASSIGNMENT - 4

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

// Define the structure of a doubly linked list node struct Node {

int data;

struct Node\* next; struct Node\* prev;

};

// Function to insert a new node at the end of the list void insertAtEnd(struct Node\*\* head, int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = data;

newNode->next = NULL; newNode->prev = NULL;

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* current = \*head; while (current->next != NULL) {

current = current->next;

}

current->next = newNode; newNode->prev = current;

}

}

// Function to add two numbers represented by linked lists

struct Node\* addNumbers(struct Node\* num1, struct Node\* num2) { struct Node\* result = NULL;

int carry = 0;

while (num1 != NULL || num2 != NULL || carry != 0) { int sum = carry;

if (num1 != NULL) { sum += num1->data; num1 = num1->next;

}

if (num2 != NULL) { sum += num2->data; num2 = num2->next;

}

carry = sum / 10; sum %= 10;

insertAtEnd(&result, sum);

}

return result;

}

// Function to reverse a doubly linked list in-place struct Node\* reverseList(struct Node\* head) {

struct Node\* current = head; struct Node\* temp = NULL;

while (current != NULL) { temp = current->prev;

current->prev = current->next; current->next = temp;

current = current->prev;

}

if (temp != NULL) { head = temp->prev;

}

return head;

}

// Function to print a linked list void printList(struct Node\* head) {

while (head != NULL) { printf("%d ", head->data); head = head->next;

}

printf("\n");

}

int main() {

// Input numbers

unsigned long long int num1 = 12365478; unsigned long long int num2 = 12685745;

// Create linked lists to represent the numbers in reverse order struct Node\* list1 = NULL;

struct Node\* list2 = NULL;

while (num1 > 0) { insertAtEnd(&list1, num1 % 10); num1 /= 10;

}

while (num2 > 0) { insertAtEnd(&list2, num2 % 10); num2 /= 10;

}

// Reverse the linked lists for proper addition list1 = reverseList(list1);

list2 = reverseList(list2);

// Add the numbers and get the result

struct Node\* result = addNumbers(list1, list2);

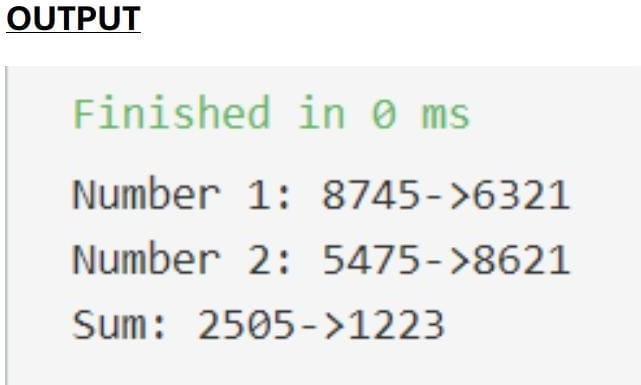
// Reverse the result for proper display result = reverseList(result);

// Print the result printf("Sum: "); printList(result);

// Free memory free(list1); free(list2); free(result);

return 0;

}



# TIME COMPLEXITY

1. CREATING LINKED LIST - O(N)

# ADDING NUMBERS - O(N)

1. REVERSING LINKED LIST - O(N)

# SPACE COMPLEXITY

1. WE USE N LISTS FOR N DIGITS HENCE SPACE COMPLEXITY - O(N)

# N = NUMBER OF DIGITS