DSA ASSIGNMENT 3  
 HARSH JAISWAL   
 BT22CSH013  
  
Q1.  
  
#include <stdio.h>

#include <stdlib.h>

struct N {

int r;

int c;

int v;

struct N\* n;

};

struct N\* C(int r, int c, int v) {

struct N\* n = (struct N\*)malloc(sizeof(struct N));

if (n == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

n->r = r;

n->c = c;

n->v = v;

n->n = NULL;

return n;

}

void D(struct N\* m[], int r, int c) {

printf("Sparse Matrix Representation:\n");

for (int i = 0; i < r; i++) {

struct N\* p = m[i];

while (p != NULL) {

printf("R: %d, C: %d, V: %d\n", p->r, p->c, p->v);

p = p->n;

}

}

}

int main() {

int r, c;

printf("Enter the number of rows and columns of the sparse matrix: ");

scanf("%d %d", &r, &c);

struct N\* m[r];

for (int i = 0; i < r; i++) {

m[i] = NULL;

}

int nz;

printf("Enter the number of non-zero elements: ");

scanf("%d", &nz);

printf("Enter the row, column, and value for each non-zero element:\n");

for (int i = 0; i < nz; i++) {

int x, y, z;

scanf("%d %d %d", &x, &y, &z);

struct N\* n = C(x, y, z);

if (m[x] == NULL) {

m[x] = n;

} else {

struct N\* p = m[x];

while (p->n != NULL) {

p = p->n;

}

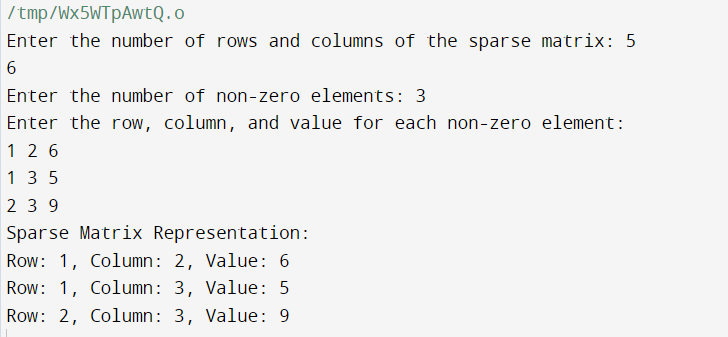
p->n = n;

}

}

D(m, r, c);

return 0;

}  
  
OUTPUT:-  
  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Q2.  
#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

typedef struct Node Node;

Node\* createNode(int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

if (newNode == NULL) {

fprintf(stderr, "Memory allocation failed.\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

Node\* addLongIntegers(Node\* num1, Node\* num2) {

Node\* result = NULL;

Node\* current = NULL;

int carry = 0;

while (num1 != NULL || num2 != NULL || carry) {

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 = sum % 10;

Node\* newNode = createNode(sum);

if (result == NULL) {

result = newNode;

current = result;

} else {

current->next = newNode;

current = current->next;

}

}

return result;

}

void displayLongInteger(Node\* num) {

if (num == NULL) {

printf("0");

return;

}

while (num != NULL) {

printf("%d", num->data);

num = num->next;

}

printf("\n");

}

int main() {

Node\* num1 = NULL;

Node\* num2 = NULL;

// Input the first long integer (represented as a linked list)

char num1Str[1000];

printf("Enter the first long integer as a string of digits: ");

scanf("%s", num1Str);

for (int i = 0; num1Str[i] != '\0'; i++) {

int digit = num1Str[i] - '0';

Node\* newNode = createNode(digit);

newNode->next = num1;

num1 = newNode;

}

// Input the second long integer (represented as a linked list)

char num2Str[1000];

printf("Enter the second long integer as a string of digits: ");

scanf("%s", num2Str);

for (int i = 0; num2Str[i] != '\0'; i++) {

int digit = num2Str[i] - '0';

Node\* newNode = createNode(digit);

newNode->next = num2;

num2 = newNode;

}

// Add the two long integers

Node\* result = addLongIntegers(num1, num2);

// Display the result

printf("Result: ");

displayLongInteger(result);

// Free memory

while (num1 != NULL) {

Node\* temp = num1;

num1 = num1->next;

free(temp);

}

while (num2 != NULL) {

Node\* temp = num2;

num2 = num2->next;

free(temp);

}

while (result != NULL) {

Node\* temp = result;

result = result->next;

free(temp);

}

return 0;

}  
  
OUTPUT:-   
