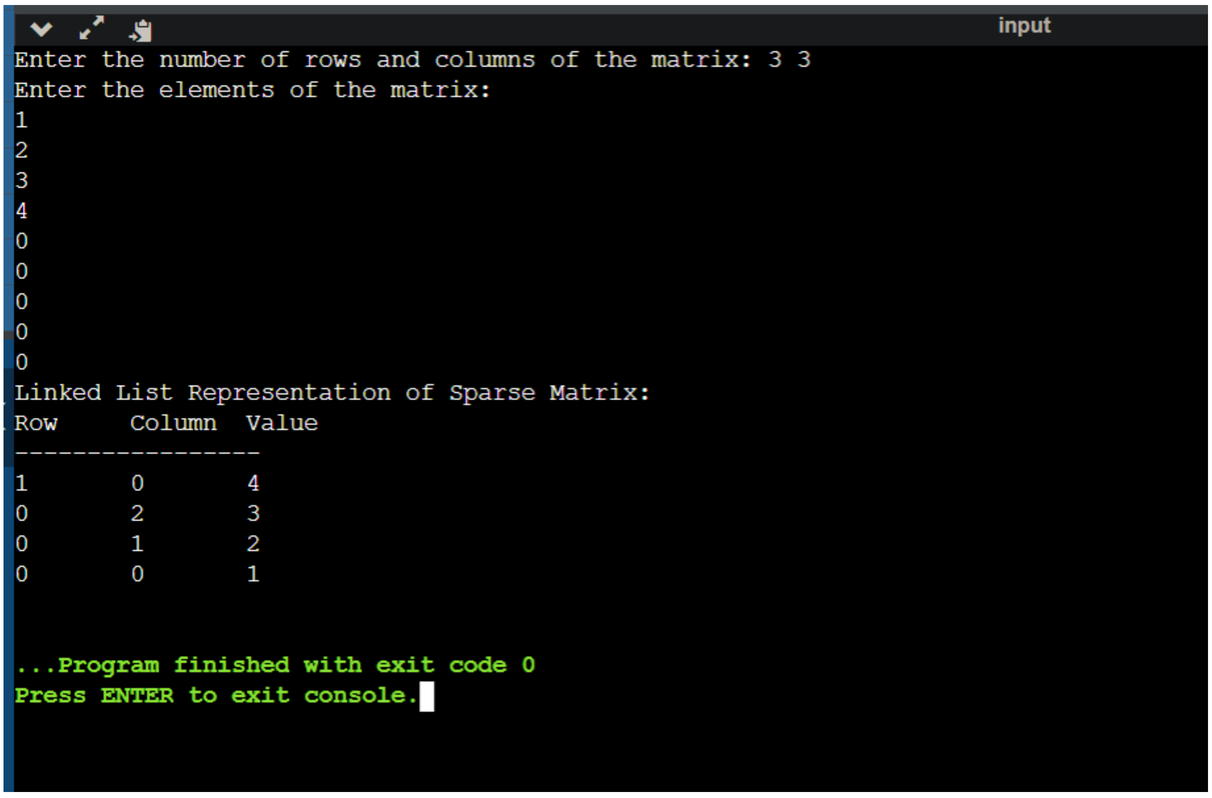
DSA LAB ASSIGNMENT 3
BT22CSH031

1)

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

struct Node {
    int row, col, value;
    struct Node* next;
};


struct Node* createNode(int row, int col, int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->row = row;
    newNode->col = col;
    newNode->value = value;
    newNode->next = NULL;
    return newNode;
}


void displaySparseMatrix(struct Node* head) {
    if (head == NULL) {
        printf("The sparse matrix is empty.\n");
        return;
    }

    printf("Row\tColumn\tValue\n");
    printf("-----------------\n");

    struct Node* current = head;
    while (current != NULL) {
        printf("%d\t%d\t%d\n", current->row, current->col, current->value);
        current = current->next;
    }
}

int main() {
    int m, n;
    printf("Enter the number of rows and columns of the matrix: ");
    scanf("%d %d", &m, &n);

    struct Node* head = NULL;

    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            int element;
            scanf("%d", &element);


            if (element != 0) {
                if (head == NULL) {
                    head = createNode(i, j, element);
                } else {
                    struct Node* newNode = createNode(i, j, element);
                    newNode->next = head;
                    head = newNode;
                }
            }
        }
    }


    printf("Linked List Representation of Sparse Matrix:\n");
    displaySparseMatrix(head);


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

    return 0;
}
```



```
Enter the number of rows and columns of the matrix: 3 3
Enter the elements of the matrix:
1
2
3
4
0
0
0
0
0
Linked List Representation of Sparse Matrix:
Row      Column   Value
------------------
1        0        4
0        2        3
0        1        2
0        0        1


...Program finished with exit code 0
Press ENTER to exit console.
```

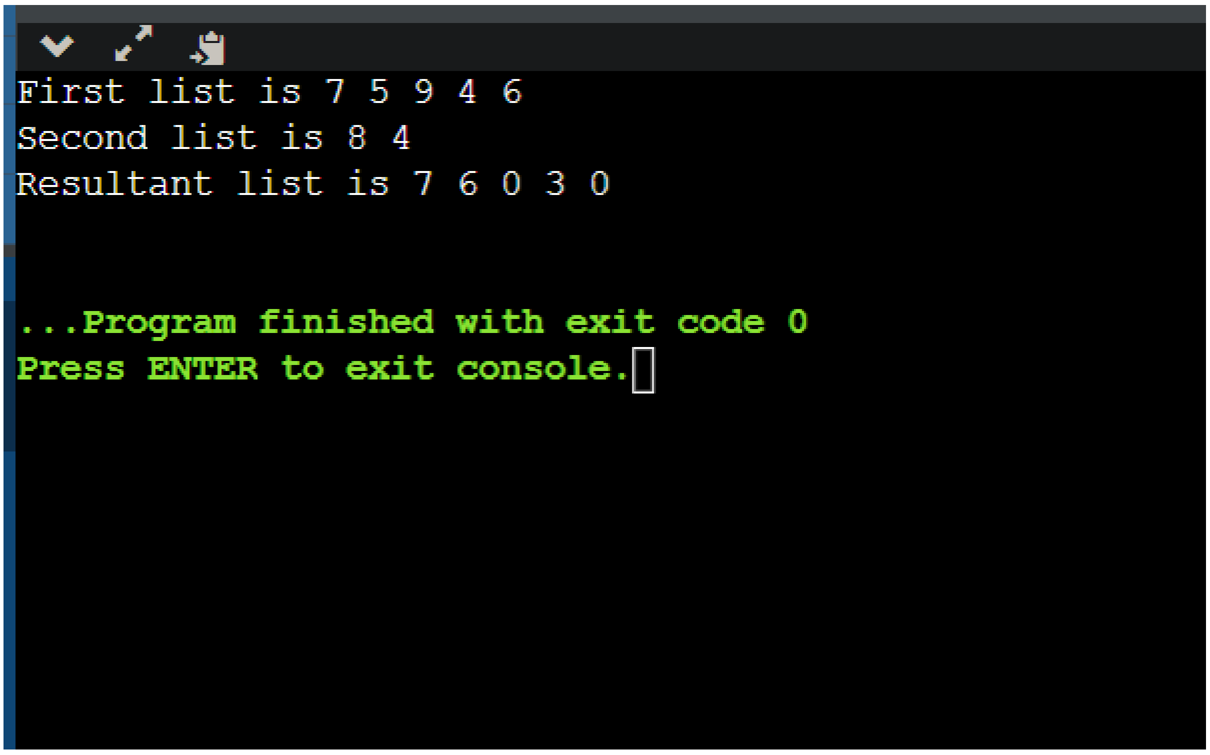
2)

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


typedef struct Node {
        int data;
        struct Node* next;
}Node;
Node* newNode(int data)
{
        Node* new_node = (Node *)malloc(sizeof(Node));
        new_node->data = data;
        new_node->next = NULL;
        return new_node;
}


void push(Node** head_ref, int new_data)
{

        Node* new_node = newNode(new_data);

        new_node->next = (*head_ref);
```

```
	(*head_ref) = new_node;
}


Node* addTwoLists(Node* first, Node* second)
{
	// res is head node of the resultant list
	Node* res = NULL;
	Node *temp, *prev = NULL;
	int carry = 0, sum;


	while (first != NULL || second != NULL) {

		sum = carry + (first ? first->data : 0) + (second ? second->data : 0);

		carry = (sum >= 10) ? 1 : 0;

		sum = sum % 10;

		temp = newNode(sum);

		if (res == NULL)
			res = temp;

		else
			prev->next = temp;


		prev = temp;


		if (first)
			first = first->next;
		if (second)
			second = second->next;
	}
	if (carry > 0)
		temp->next = newNode(carry);

	return res;
}

Node* reverse(Node* head)
{
	if (head == NULL || head->next == NULL)
		return head;
	// reverse the rest list and put the first element at the end
```

```
        Node* rest = reverse(head->next);
        head->next->next = head;
        head->next = NULL;
        // fix the head pointer
        return rest;
}


void printList(Node* node)
{
        while (node != NULL) {
                printf("%d ",node->data);
                node = node->next;
        }
        printf("\n");
}


int main(void)
{
        Node* res = NULL;
        Node* first = NULL;
        Node* second = NULL;


        push(&first, 6);
        push(&first, 4);
        push(&first, 9);
        push(&first, 5);
        push(&first, 7);
        printf("First list is ");
        printList(first);


        push(&second, 4);
        push(&second, 8);
        printf("Second list is ");
        printList(second);


        first = reverse(first);
        second = reverse(second);

        res = addTwoLists(first, second);


        res = reverse(res);
        printf("Resultant list is ");
```

```
		printList(res);
		return 0;
}
```



```
First list is 7 5 9 4 6
Second list is 8 4
Resultant list is 7 6 0 3 0


...Program finished with exit code 0
Press ENTER to exit console.
```