```c
// Name: Ayush Gupta
// Roll No. : MNS2025027

// 1. Take a matrix as an input from the user such that
the number of zeros are more than non-zero
// elements. Store and Print it. Now store this matrix as
a sparse matrix using a linked list.
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int r, c, v;
    struct node* next;
};
struct node* create(int r, int c, int v)
{
    struct node* t = malloc(sizeof(struct node));
    t->r = r;
    t->c = c;
    t->v = v;
    t->next = NULL;
    return t;
}
void display(struct node* h)
{
    while (h)
    {
        printf("%d %d %d\n", h->r, h->c, h->v);
        h = h->next;
    }
}
int main()
```

```c
{
    int m, n, i, j;
    printf("Enter rows and cols: ");
    scanf("%d%d", &m, &n);
    int a[m][n];
    printf("Enter matrix:\n");
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++) scanf("%d", &a[i][j]);
    printf("Matrix:\n");
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++) printf("%d ", a[i][j]);
        printf("\n");
    }
    struct node *head = NULL, *last = NULL;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (a[i][j] != 0)
            {
                struct node* t = create(i, j, a[i][j]);
                if (!head)
                    head = last = t;
                else
                {
                    last->next = t;
                    last = t;
                }
            }
    printf("Sparse form:\n");
    display(head);
    return 0;
}
```

```
// 2. Suppose the user wants to change a zero to a
non-zero element or vice-versa. Create the
// function(s) to update the sparse matrix that is
already created and stored in a linked list.
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int r, c, v;
    struct node* next;
};
struct node* create(int r, int c, int v)
{
    struct node* t = malloc(sizeof(struct node));
    t->r = r;
    t->c = c;
    t->v = v;
    t->next = NULL;
    return t;
}
void update(struct node** h, int r, int c, int v)
{
    struct node *p = *h, *prev = NULL;
    while (p)
    {
        if (p->r == r && p->c == c)
        {
            if (v == 0)
            {
                if (prev)
                    prev->next = p->next;
```

```c
                else
                    *h = p->next;
                free(p);
            }
            else
                p->v = v;
            return;
        }
        prev = p;
        p = p->next;
    }
    if (v != 0)
    {
        struct node* t = create(r, c, v);
        t->next = *h;
        *h = t;
    }
}
void display(struct node* h)
{
    while (h)
    {
        printf("%d %d %d\n", h->r, h->c, h->v);
        h = h->next;
    }
}
int main()
{
    int n, i, r, c, v;
    struct node* head = NULL;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d%d%d", &r, &c, &v);
        struct node* t = create(r, c, v);
        t->next = head;
        head = t;
    }
    printf("Enter position and value to update: ");
    scanf("%d%d%d", &r, &c, &v);
    update(&head, r, c, v);
    display(head);
    return 0;
}

// 3. Create a program to add two sparse matrices.
Assuming that you have created two sparse
// matrices using linked lists as in program 1.
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int r, c, v;
    struct node* next;
};
struct node* create(int r, int c, int v)
{
    struct node* t = malloc(sizeof(struct node));
    t->r = r;
    t->c = c;
    t->v = v;
    t->next = NULL;
    return t;
```

```c
}
void insert(struct node** h, int r, int c, int v)
{
    if (v == 0) return;
    struct node *p = *h, *prev = NULL;
    while (p && (p->r < r || (p->r == r && p->c < c)))
    {
        prev = p;
        p = p->next;
    }
    if (p && p->r == r && p->c == c)
    {
        p->v += v;
        if (p->v == 0)
        {
            if (prev)
                prev->next = p->next;
            else
                *h = p->next;
            free(p);
        }
        return;
    }
    struct node* t = create(r, c, v);
    t->next = p;
    if (prev)
        prev->next = t;
    else
        *h = t;
}
struct node* add(struct node* a, struct node* b)
{
    struct node* res = NULL;
    while (a)
    {
        insert(&res, a->r, a->c, a->v);
        a = a->next;
    }
    while (b)
    {
        insert(&res, b->r, b->c, b->v);
        b = b->next;
    }
    return res;
}
void display(struct node* h)
{
    while (h)
    {
        printf("%d %d %d\n", h->r, h->c, h->v);
        h = h->next;
    }
}
int main()
{
    int n, i, r, c, v;
    struct node *a = NULL, *b = NULL;
    printf("Enter number of elements in first matrix: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d%d%d", &r, &c, &v);
        insert(&a, r, c, v);
    }
    printf("Enter number of elements in second matrix: 
");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d%d%d", &r, &c, &v);
        insert(&b, r, c, v);
    }
    struct node* sum = add(a, b);
    printf("Sum:\n");
    display(sum);
    return 0;
}
```