```c
//Sparse Matrix operations
#include <stdio.h>
#include <string.h>

struct sparse
{
    int row,col;
    int value;
};

void matrixify(struct sparse spa_mat[])
{
    printf("Normal Matrix : \n");
    int mat[spa_mat[0].row][spa_mat[0].col];
    memset(mat, 0, sizeof(mat[0][0]) * spa_mat[0].row * spa_mat[0].col);
    for(int i=1 ; i<=spa_mat[0].value ; i++)
    {
        mat[spa_mat[i].row][spa_mat[i].col] = spa_mat[i].value;
    }
    for(int i=0 ; i<spa_mat[0].row ; i++)
    {
        for(int j=0 ; j<spa_mat[0].col ; j++)
        {
            printf("%6d", mat[i][j]);
        }
        printf("\n");
    }

}

void sparsify(int m,int n, int mat[10][10])
{
    printf("Sparse Matrix : \n");
    struct sparse spa_mat[50];
    spa_mat[0].row = m;
    spa_mat[0].col = n;
    int k=1;
    for(int i=0 ; i<m ; i++)
    {
        for(int j=0 ; j<n ; j++)
        {
            if(mat[i][j] != 0)
            {
                spa_mat[k].row = i;
                spa_mat[k].col = j;
                spa_mat[k].value = mat[i][j];
                k++;
            }
        }
    }
    spa_mat[0].value = k-1;
    for(int i=0 ; i<k ; i++)
    {
        printf("%d %6d %6d\n", spa_mat[i].row, spa_mat[i].col, spa_mat[i].value);
    }
    matrixify(spa_mat);
}

void input_spa_mat(struct sparse mat[])
{
    printf("m : ");
    scanf("%d", &mat[0].row);
    printf("n : ");
    scanf("%d", &mat[0].col);
    printf("k : ");
    scanf("%d", &mat[0].value);
    for(int i=1 ; i<= mat[0].value ; i++)
    {
        scanf("%d %d %d", &mat[i].row, &mat[i].col, &mat[i].value);
    }
}

void print_spa_mat(struct sparse mat[])
{
    for(int i=0 ; i<=mat[0].value ; i++)
    {
        printf("%d %6d %6d\n", mat[i].row, mat[i].col, mat[i].value);
    }
}

void add_sparse(struct sparse a[], struct sparse b[])
{
    int i,j,k;
    struct sparse c[50];
    c[0].value = 0;
    c[0].row = a[0].row;
    c[0].col = a[0].col;
    for(i=1,j=1,k=1 ; i<=a[0].value && j<=b[0].value ; )
    {
        if(a[i].row == b[j].row)
        {
            if(a[i].col == b[j].col)
            {
                c[k].row = a[i].row;
                c[k].col = a[i].col;
                c[k].value = a[i].value + b[j].value;
                c[0].value++;
                i++;
                j++;
                k++;
            }
            else if(a[i].col < b[j].col)
            {
                c[k].row = a[i].row;
                c[k].col = a[i].col;
                c[k].value = a[i].value;
                c[0].value++;
                i++;
                k++;
            }
            else
            {
                c[k].row = b[j].row;
                c[k].col = b[j].col;
                c[k].value = b[j].value;
                c[0].value++;
                j++;
                k++;
            }
        }
        else if(a[i].row < b[j].row)
        {
            c[k].row = a[i].row;
            c[k].col = a[i].col;
            c[k].value = a[i].value;
            c[0].value++;
            i++;
            k++;
        }
        else
        {
            c[k].row = b[j].row;
            c[k].col = b[j].col;
            c[k].value = b[j].value;
            c[0].value++;
            j++;
            k++;
        }
    }
    if(i<=a[0].value)
    {
        for(; i<=a[0].value ;)
        {
            c[k].row = a[i].row;
            c[k].col = a[i].col;
            c[k].value = a[i].value;
            c[0].value++;
            i++;
            k++;
        }
    }
    if(j<=b[0].value)
    {
        for(; i<=b[0].value ;)
        {
            c[k].row = b[j].row;
            c[k].col = b[j].col;
            c[k].value = b[j].value;
            c[0].value++;
            j++;
            k++;
        }
    }
    printf("\n");
    print_spa_mat(c);
}

void transpose_sparse(struct sparse s[], struct sparse s_trans[])
{
    s_trans[0].row = s[0].col;
    s_trans[0].col = s[0].row;
    s_trans[0].value = s[0].value;
    int k=1;
    if(s_trans[0].value > 0)
    {
        for (int i = 0; i < s[0].col; ++i)
        {
            for (int j = 1; j <= s[0].value; ++j)
            {
                if(s[j].col == i)
                {
                    s_trans[k].row = s[j].col;
                    s_trans[k].col = s[j].row;
                    s_trans[k].value = s[j].value;
                    k++;
                }
            }
        }
    }
}

int main()
{
   struct sparse a[50], b[50], c[50];
   printf("Input Matrix 1 \n");
   input_spa_mat(a);
   printf("Input Matrix 2 \n");
   input_spa_mat(b);
   printf("Matrix Addition\n");
   add_sparse(a,b);
   transpose_sparse(a,c);
   printf("Transpose of Matrix 1\n");
   print_spa_mat(c);
}
/*
OUTPUT
```

```
Input Matrix 1
m : 3
n : 3
k : 2
0 0 1
2 2 2
Input Matrix 2
m : 3
n : 3
k : 2
1 1 1
1 0 1
Matrix Addition
3      3      4
0      0      1
1      1      1
1      0      1
2      2      2
Transpose of Matrix 1
3      3      2
0      0      1
2      2      2
*/
```