```c
#include <stdio.h>

void inputMatrix(int matrix[100][100], int *rows, int *cols);
void convertToSparse(int matrix[100][100], int rows, int cols, int
sparse[100][3], int *count);
void addSparseMatrices(int T1[100][3], int n1, int T2[100][3], int n2,
int R[100][3], int *nR);
void printSparseMatrix(int sparse[100][3], int count);

int main() {
    int a[100][100], b[100][100];
    int T1[100][3], T2[100][3], R[100][3];
    int r, c, r1, c1;
    int n1 = 0, n2 = 0, nR = 0;

    printf("Enter the number of rows and columns of the First
matrix:\n");
    inputMatrix(a, &r, &c);
    printf("Enter the number of rows and columns of the Second
matrix:\n");
    inputMatrix(b, &r1, &c1);

    if (r != r1 || c != c1) {
        printf("Error: Matrices must be of the same dimensions for
addition.\n");
        return 1;
    }

    convertToSparse(a, r, c, T1, &n1);
    convertToSparse(b, r1, c1, T2, &n2);

    addSparseMatrices(T1, n1, T2, n2, R, &nR);

    printf("The Sparse matrix representation of the Resultant matrix
is:\n");
    printf("Row\tColumn\tValue\n");
    printSparseMatrix(R, nR);

    return 0;
}

void inputMatrix(int matrix[100][100], int *rows, int *cols) {
    scanf("%d %d", rows, cols);
    printf("Enter the matrix elements:\n");
    for (int i = 0; i < *rows; i++) {
        for (int j = 0; j < *cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

void convertToSparse(int matrix[100][100], int rows, int cols, int
sparse[100][3], int *count) {
    int k = 1;
    *count = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] != 0) {
```

```
                sparse[k][0] = i;
                sparse[k][1] = j;
                sparse[k][2] = matrix[i][j];
                k++;
                (*count)++;
            }
        }
    }
    sparse[0][0] = rows;
    sparse[0][1] = cols;
    sparse[0][2] = *count;
}

void addSparseMatrices(int T1[100][3], int n1, int T2[100][3], int n2,
int R[100][3], int *nR) {
    int k = 1, p1 = 1, p2 = 1;
    *nR = 0;

    while (p1 <= n1 && p2 <= n2) {
        if (T1[p1][0] == T2[p2][0] && T1[p1][1] == T2[p2][1]) {
            R[k][0] = T1[p1][0];
            R[k][1] = T1[p1][1];
            R[k][2] = T1[p1][2] + T2[p2][2];
            p1++;
            p2++;
        } else if (T1[p1][0] < T2[p2][0] || (T1[p1][0] == T2[p2][0] &&
T1[p1][1] < T2[p2][1])) {
            R[k][0] = T1[p1][0];
            R[k][1] = T1[p1][1];
            R[k][2] = T1[p1][2];
            p1++;
        } else {
            R[k][0] = T2[p2][0];
            R[k][1] = T2[p2][1];
            R[k][2] = T2[p2][2];
            p2++;
        }
        k++;
    }

    while (p1 <= n1) {
        R[k][0] = T1[p1][0];
        R[k][1] = T1[p1][1];
        R[k][2] = T1[p1][2];
        p1++;
        k++;
    }

    while (p2 <= n2) {
        R[k][0] = T2[p2][0];
        R[k][1] = T2[p2][1];
        R[k][2] = T2[p2][2];
        p2++;
        k++;
    }

    R[0][0] = T1[0][0];
    R[0][1] = T1[0][1];
```

```
    R[0][2] = (k - 1);

    *nR = k;
}

void printSparseMatrix(int sparse[100][3], int count) {
    for (int i = 0; i < count; i++) {
        printf("%d\t%d\t%d\n", sparse[i][0], sparse[i][1], sparse[i][2]);
    }
}
```