#include <iostream>

using namespace std;

// Structure to represent a non-zero element in the sparse matrix

struct Element {

int row;

int col;

int value;

};

// Structure to represent a sparse matrix

struct SparseMatrix {

int numRows;

int numCols;

int numNonZero;

Element\* elements;

};

// Function to initialize a sparse matrix

SparseMatrix initializeSparseMatrix(int numRows, int numCols, int numNonZero) {

SparseMatrix matrix;

matrix.numRows = numRows;

matrix.numCols = numCols;

matrix.numNonZero = numNonZero;

// Dynamic allocation for the array of non-zero elements

matrix.elements = new Element[numNonZero];

// Initialize each element with zeros initially

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

matrix.elements[i].row = 0;

matrix.elements[i].col = 0;

matrix.elements[i].value = 0;

}

return matrix;

}

// Function to set a non-zero element in the sparse matrix

void setElement(SparseMatrix& matrix, int row, int col, int value) {

// Check if the provided row and column are within the matrix dimensions

if (row >= 0 && row < matrix.numRows && col >= 0 && col < matrix.numCols) {

// Find the index of the element in the array

int index = row \* matrix.numCols + col;

// Set the values for the non-zero element

matrix.elements[index].row = row;

matrix.elements[index].col = col;

matrix.elements[index].value = value;

} else {

cout << "Invalid row or column." << endl;

}

}

// Function to display the sparse matrix

void displaySparseMatrix(const SparseMatrix& matrix) {

for (int i = 0; i < matrix.numNonZero; ++i) {

cout << "Row: " << matrix.elements[i].row

<< ", Column: " << matrix.elements[i].col

<< ", Value: " << matrix.elements[i].value

<< endl;

}

}

// Function to deallocate memory used by the sparse matrix

void deallocateSparseMatrix(SparseMatrix& matrix) {

delete[] matrix.elements;

}

int main() {

// Example usage

SparseMatrix mySparseMatrix = initializeSparseMatrix(3, 3, 4);

setElement(mySparseMatrix, 0, 1, 5);

setElement(mySparseMatrix, 1, 2, 3);

setElement(mySparseMatrix, 2, 0, 1);

setElement(mySparseMatrix, 2, 2, 9);

displaySparseMatrix(mySparseMatrix);

// Deallocate memory

deallocateSparseMatrix(mySparseMatrix);

return 0;