include <iostream>

// Define a node structure to represent non-zero elements in the matrix

struct Node {

int row;

int col;

int value;

Node\* next;

};

// Define a LinkedList class to represent a sparse matrix using linked list

class LinkedList {

public:

LinkedList() : head(nullptr) {}

// Function to insert a new node into the linked list

void insertNode(int row, int col, int value) {

Node\* newNode = new Node{row, col, value, nullptr};

if (!head || (row < head->row) || (row == head->row && col < head->col)) {

newNode->next = head;

head = newNode;

} else {

Node\* current = head;

while (current->next && (row > current->next->row || (row == current->next->row && col > current->next->col))) {

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

}

// Function to print the linked list

void printList() {

Node\* current = head;

while (current) {

std::cout << "(" << current->row << ", " << current->col << "): " << current->value << " ";

current = current->next;

}

std::cout << std::endl;

}

private:

Node\* head;

};

// Define a SparseMatrix class that uses the LinkedList to represent the entire matrix

class SparseMatrix {

public:

SparseMatrix(int rows, int cols) : rows(rows), cols(cols), matrix(new LinkedList[rows]) {}

// Function to insert a new element into the matrix

void insertElement(int row, int col, int value) {

if (row < 0 || row >= rows || col < 0 || col >= cols) {

std::cerr << "Invalid matrix indices!" << std::endl;

return;

}

matrix[row].insertNode(row, col, value);

}

// Function to print the entire sparse matrix

void printMatrix() {

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

if (matrix[i].head) {

matrix[i].printList();

}

}

}

private:

int rows;

int cols;

LinkedList\* matrix;

};

int main() {

// Example usage

SparseMatrix sparseMat(3, 4);

// Inserting elements into the sparse matrix

sparseMat.insertElement(0, 1, 5);

sparseMat.insertElement(1, 2, 8);

sparseMat.insertElement(2, 0, 3);

sparseMat.insertElement(2, 3, 2);

// Printing the sparse matrix

std::cout << "Sparse Matrix:" << std::endl;

sparseMat.printMatrix();