#include <iostream>

// Node structure to represent each element in the sparse matrix

struct Node {

int row, col, value;

Node\* next;

Node(int r, int c, int v) : row(r), col(c), value(v), next(nullptr) {}

};

// LinkedList class for sparse matrix

class SparseMatrixLinkedList {

private:

Node\* head; // Pointer to the head of the linked list

public:

// Constructor

SparseMatrixLinkedList() : head(nullptr) {}

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

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

Node\* newNode = new Node(row, col, value);

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

// Insert at the beginning

newNode->next = head;

head = newNode;

} else {

// Insert in the middle or at the end

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 display the linked list

void display() {

Node\* current = head;

while (current) {

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

current = current->next;

}

std::cout << std::endl;

}

// Destructor to free memory

~SparseMatrixLinkedList() {

Node\* current = head;

while (current) {

Node\* next = current->next;

delete current;

current = next;

}

}

};

int main() {

// Example usage

SparseMatrixLinkedList matrix;

matrix.insert(1, 1, 5);

matrix.insert(0, 2, 3);

matrix.insert(2, 0, 8);

matrix.display();

return 0;

}