#include <iostream>

struct Node {

int row;

int col;

int value;

Node\* next;

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

};

class SparseMatrix {

private:

int rows, cols;

Node\* head;

public:

SparseMatrix(int r, int c) : rows(r), cols(c), head(nullptr) {}

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

if (value == 0) return;

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

if (!head) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next) {

temp = temp->next;

}

temp->next = newNode;

}

}

int getElement(int row, int col) {

Node\* temp = head;

while (temp) {

if (temp->row == row && temp->col == col) {

return temp->value;

}

temp = temp->next;

}

return 0;

}

void display() {

Node\* temp = head;

int sparse\_matrix[rows][cols] = {0};

while (temp) {

sparse\_matrix[temp->row][temp->col] = temp->value;

temp = temp->next;

}

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

for (int j = 0; j < cols; ++j) {

std::cout << sparse\_matrix[i][j] << " ";

}

std::cout << std::endl;

}

}

void displaySparse() {

Node\* temp = head;

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

while (temp) {

std::cout << "Row: " << temp->row << ", Column: " << temp->col << ", Value: " << temp->value << std::endl;

temp = temp->next;

}

}

~SparseMatrix() {

Node\* temp = head;

while (temp) {

Node\* next = temp->next;

delete temp;

temp = next;

}

}

};

int main() {

SparseMatrix sparseMatrix(4, 5);

sparseMatrix.addElement(0, 2, 3);

sparseMatrix.addElement(0, 4, 4);

sparseMatrix.addElement(1, 2, 5);

sparseMatrix.addElement(1, 3, 7);

sparseMatrix.addElement(3, 1, 2);

sparseMatrix.addElement(3, 2, 6);

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

sparseMatrix.display();

std::cout << "\nSparse Matrix Representation:" << std::endl;

sparseMatrix.displaySparse();

std::cout << "\nElement at (1, 2): " << sparseMatrix.getElement(1, 2) << std::endl;

std::cout << "Element at (0, 0): " << sparseMatrix.getElement(0, 0) << std::endl;

return 0;

}