```
#include template class ClassArray { private: T* array; int capacity; int current; public: ClassArray() : capacity(10), current(0) { array = new T[capacity]; } ~ClassArray() { delete[] array; } void add(T item) { if (current == capacity) { T* temp = new T[capacity * 2]; for (int i = 0; i < capacity; i++) { temp[i] = array[i]; } delete[] array; array = temp; capacity *= 2; } array[current++] = item; } T& get(int index) { if (index < 0 || index >= current) { std::cerr << "Index out of bounds!" << std::endl; return array[0]; // Return first element as a fallback } return array[index]; } void set(int index, T value) { if (index >= 0 && index < current) { array[index] = value; } else { std::cerr << "Index out of bounds!" << std::endl; } } int size() const { return current; } }; template class VectorClass { private: ClassArray arr; public: void push(T item) { arr.add(item); } T& at(int index) { return arr.get(index); } void set(int index, T item) { arr.set(index, item); // Use the set method of ClassArray } int size() const { return arr.size(); } }; template class Matrix { private: VectorClass*> rows; public: Matrix(int numRows, int numCols, T initialValue) { for (int i = 0; i < numRows; i++) { VectorClass* row = new VectorClass; for (int j = 0; j < numCols; j++) { row->push(initialValue); } rows.push(row); } } ~Matrix() { for (int i = 0; i < rows.size(); i++) { delete rows.at(i); } } void set(int row, int col, T value) { if (row < 0 || row >= rows.size()) { std::cerr << "Row index out of bounds!" << std::endl; return; } if (col < 0 || col >= rows.at(row)->size()) { std::cerr << "Column index out of bounds!" << std::endl; return; } rows.at(row)->set(col, value); } T get(int row, int col) { if (row < 0 || row >= rows.size() || col < 0 || col >= rows.at(row)->size()) { std::cerr << "Index out of bounds!" << std::endl; return rows.at(0)->at(0); } return rows.at(row)->at(col); } int numRows() const { return rows.size(); } int numCols() { if (rows.size() > 0) return (rows.at(0)->size()); else return 0; } }; int main() { Matrix myMatrix(3, 3, 0); // Create a 3x3 matrix initialized with 0s std::cout << "Initial Matrix:" << std::endl; for (int i = 0; i < myMatrix.numRows(); i++) { for (int j = 0; j < myMatrix.numCols(); j++) { std::cout << myMatrix.get(i, j) << " "; } std::cout << std::endl; } // Set the middle element to 5 myMatrix.set(1, 1, 5); std::cout << "Updated Matrix (1,1 set to 5):" << std::endl; for (int i = 0; i < myMatrix.numRows(); i++) { for (int j = 0; j < myMatrix.numCols(); j++) { std::cout << myMatrix.get(i, j) << " "; } std::cout << std::endl; } // Displaying the element at position (1,1) std::cout << "Element at (1,1): " << myMatrix.get(1, 1) << std::endl; // Display the total number of rows and columns std::cout << "Number of rows: " << myMatrix.numRows() << std::endl; std::cout << "Number of columns: " << myMatrix.numCols() << std::endl; return 0; }
```