```cpp
1   /*
2   --------------------------------------------------------------------------------
3   File name     : matrixUtilities.cpp
4   Authors       : Miguel Jalube, Leandro Saraiva Maia
5   Creation date : 07.12.2021
6
7   Description   : See matrixUtilities.h
8   Remark(s)     :
9
10  Compiler      : Mingw-w64 g++ 11.1.0
11  --------------------------------------------------------------------------------
12  */
13
14  #include "matrixUtilities.h" // Prototypes and aliases (Matrix, VInt)
15  #include <vector>             // required for internal use of vector (Matrix, VInt)
16  #include <algorithm>          // required for sort functions
17  #include <ctime>              // required for random seed
18  #include <numeric>            // required for accumulate
19  #include <iterator>           // required for distance()
20  #include <iostream>           // required for cout
21  #include <ctime>              // required to generate a seed based on time
22  #include <random>             // required to generate random numbers
23
24  using namespace std;
25
26  // ----- predicates -----
27  /**
28   * Checks if 2 vectors have the same size
29   * @param vector1
30   * @param vector2
31   * @return boolean true if vectors have the same size
32   */
33  bool isSameSizeAs(const VInt& vector1, const VInt& vector2) {
34      return vector1.size() == vector2.size();
35  }
36
37  /**
38   * Checks if vector1 smaller than vector2
39   * @param vector1
40   * @param vector2
41   * @return boolean true if vector1 is smaller
42   */
43  bool isSmallerThan(const VInt& vector1, const VInt& vector2){
44      return vector1.size() < vector2.size();
45  }
46
47  /**
48   * Checks if the first element of the vector1 is smaller than the first element of
49   * the vector2. When one of the two vectors is empty, we sort it first
50   * @param vector1
51   * @param vector2
52   * @return boolean true if the first value of the vector1 is smaller
53   */
54  bool isFirstElemSmallerThan(const VInt& vector1, const VInt& vector2) {
55      return (vector1.empty() || vector2.empty()) || vector1.front() < vector2.front();
56  }
57
58  // ----- utilities functions -----
59  bool isSquare(const Matrix& matrix){
60      // Returns true if the matrix is empty, otherwise checks if the matrix is
61      // regular AND if the size of a vector (since they are regular they have all
62      // the same size, so we can take any vector) is equal to the size of the matrix
63      return matrix.empty() ||
64              (isRegular(matrix) && matrix.front().size() == matrix.size());
65  }
66
67  bool isRegular(const Matrix& matrix){
68      // Returns true if the matrix is empty, otherwise checks if every element has
69      // the same size as the next one. If this is true, then it means that every
70      // vector has the same size
71      return matrix.empty() || equal(matrix.begin(), matrix.end() - 1,
72                                      matrix.begin() + 1, isSameSizeAs);
```

```cpp
 73    }
 74
 75    // ! This function is named "minCol" in the exercice, but we renamed it "minRow".
 76    // We did it because it did not reflect what the function actually does
 77    size_t minRow(const Matrix& matrix){
 78        // Return 0 if matrix is empty, otherwise compare the size of every row and
 79        // returns the size of the smallest
 80        return matrix.empty() ? 0 :
 81                (*min_element(matrix.begin(), matrix.end(), isSmallerThan)).size();
 82    }
 83
 84    VInt sumRow(const Matrix& matrix){
 85        if (matrix.empty()) return {};
 86
 87        VInt result = VInt(matrix.size());
 88        // For each vector in the matrix, we sum the elements of the vector and put it
 89        // in a vector that is indexed in the same way as matrix (that is why we use a
 90        // "for int i" and not a "for auto i :", because we want the int index).
 91        for (size_t i = 0; i < matrix.size(); ++i) {
 92            result[i] = accumulate(matrix[i].begin(), matrix[i].end(), 0);
 93        }
 94
 95        return result;
 96    }
 97
 98    VInt sumColumn(const Matrix& matrix){
 99        VInt result;
100        if(!matrix.empty()) {
101            // Resizes result vector to the same size as the biggest vector in the matrix
102            result.resize(
103                (*max_element(matrix.begin(), matrix.end(), isSmallerThan)).size());
104
105            // For each vector in the matrix increment the result vector (default value
106            // is 0) adding the value of matrix[i][j] to result[j]
107            for (const VInt &i: matrix) {
108                for (size_t j = 0; j < i.size(); ++j) {
109                    result.at(j) += i.at(j);
110                }
111            }
112        }
113        return result;
114    }
115
116    VInt vectSumMin(const Matrix& matrix){
117        if (matrix.empty()) return {};
118
119        // First, get a vector containing the sum of each row (indexed identically as
120        // the matrix).
121        VInt vSumRow = sumRow(matrix);
122        // Then, we get index of the row with the smallest sum by getting the distance
123        // between the being iterator and the min_element iterator.
124        size_t indexOfMinSumRow = (size_t)distance(vSumRow.begin(),
125                                    min_element(vSumRow.begin(), vSumRow.end()));
126        // Since vSumRow is indexed identically as the matrix, we can access the matrix
127        // with the same index
128        return matrix[indexOfMinSumRow];
129    }
130
131    void shuffleMatrix(Matrix& matrix){
132        if(!matrix.empty()) {
133            // Sets the random seed in static, so it is not reset for each call
134            static long long int seed = time(nullptr);
135
136            // Shuffles the matrix elements using the random generator mt19937 seeded using
137            // the ctime library
138            shuffle(matrix.begin(), matrix.end(), mt19937(seed));
139        }
140    }
141
142    void sortMatrix(Matrix& matrix){
143        if (matrix.empty()) return;
144
```

```cpp
145        // Sort matrix by looking at the smallest first element of each row.
146        sort(matrix.begin(), matrix.end(), isFirstElemSmallerThan);
147    }
148
149    ostream& operator<< (ostream& os,const VInt& vector){
150        os << "(";
151        // Loop to send in the ostream, the value at *i
152        for(VInt::const_iterator i = vector.begin(); i != vector.end(); ++i){
153            if(i != vector.begin()){
154                os << ", ";
155            }
156            os << *i;
157        }
158        os << ")";
159        return os;
160    }
161
162    ostream& operator<< (ostream& os,const Matrix& matrix){
163        os << "[";
164        // Loop to send in the ostream, the vector at *i
165        for(Matrix::const_iterator i = matrix.begin(); i != matrix.end(); ++i){
166            if(i != matrix.begin()){
167                os << ", ";
168            }
169            os << *i;
170        }
171        os << "]";
172        return os;
173    }
```