Math 173: Object Oriented Programming

Project 4: Matrix Operations

This project requires you to implement a class for square matrices with real entries. (Recall that a square matrix is a matrix with the same number of rows as columns.) You will define the class **SquareMatrix** and provide member functions for elementary row operations. You will also overload the equality and stream operators as described below.

Constructors

There are four required constructors which work as follows.

- Initialize a SquareMatrix with the number of rows.
  ```
  SquareMatrix M(5);
  ```
  (A 5×5 matrix of zeros.)
- Initialize with an initializer list (`<initializer_list>`).
  ```
  SquareMatrix M = {1,2,3,4,5,6,7,8,9};
  ```
  (This is a 3×3 matrix.)
- Initialize with a vector and the number of rows.
  ```
  std::vector numbers = {1,2,3,4,5,6,7,8,9,10.34,-17.32};
  SquareMatrix N(numbers,2);
  ```
  (The matrix N is $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.)
- Initialize with an `istream` (such as `cin` or an `ifstream` object) and the number of rows. For example, if using an `ifstream`, the associated file should contain whitespace-separated data such as

  $$3.4 \quad -5.6 \quad 34 \quad 4.5$$

  If the file containing this data is *data.txt* and
  ```
  std::ifstream infile("data.txt");
  ```
  is successful, then
  ```
  SquareMatrix P(infile,2);
  ```
  represents the matrix $\begin{bmatrix} 3.4 & -5.6 \\ 34 & 4.5 \end{bmatrix}$.

Store the entries of the matrix in a `vector<double>`. Assign values to the matrix with row-major ordering, as demonstrated above. If the constructor requires the number of

rows to be given, and there are too few objects provided, throw an exception. If there are more than enough objects provided to the constructor, ignore the extras. However, the constructor which takes an initializer list must throw an exception if the number of objects given is not a perfect square.

Stream Operators

You must overload `operator<<` and `operator>>` for output and input. I don't care about how you format the output, as long as each row is on a separate line. For example,

```
std::vector<double> data = {1,0,0,2,2,2,3,3,3,4};
SquareMatrix M(data,2), N(data,3), P={1,1,0,1};
std::cout << M << N << P;
```

might be formatted as

```
1  0
0  2

1  0  0
2  2  2
3  3  3

1  1
0  1
```

The input stream must work with an already existing matrix, and, if successful, replace the entries of the matrix with the input. For example,

```
SquareMatrix M(2), N={5,6,7,8};
std::cin >> M >> N; // User enters 1 -2 3.5 7 0 0 0 0
std::cout << M << N;
```

should show

```
1  -2
3.5  7

0  0
0  0
```

Row Operations

The class must include member functions for elementary row operations. Recall that if X and Y are rows of a matrix M, the elementary row operations are

1. Swap two rows. ($X \leftrightarrow Y$)
2. Multiply a row by a nonzero scalar α. ($X \leftarrow \alpha \cdot X$)
3. Replace a row by its sum with a scalar times another row. ($X \leftarrow X + \alpha \cdot Y$)

Member functions for these operations must have the following names and arguments. Row numbers are used to represent the rows. **Important:** The first row is numbered 1.

1. `swap(unsigned int xrow, unsigned int yrow)`
2. `scale(double scalar, unsigned int row)`
3. `xpay(unsigned int xrow, double scalar, unsigned int yrow)`

For example,

```
SquareMatrix M = {1,2,3,2,3,4,-1,0,2,3,2,-2,4,0,0,1};
M.swap(2,3); // Swap 2nd and 3rd rows
M.xpay(1,0.5,4); // R1 <- R1 + (0.5)*R4
M.scale(-1.5,3); // R3 <- (-1.5)*R3
```

results in

$$M = \begin{bmatrix} 3 & 2 & 3 & 2.5 \\ 2 & 3 & 2 & -2 \\ -4.5 & -6 & 1.5 & 0 \\ 4 & 0 & 0 & 1 \end{bmatrix}$$

More Operators

Lastly, implement `operator==` and `operator!=` for your SquareMatrix class.

```
SquareMatrix M(3), N(3), P(4);
M == N; // true
M == P; // false
M != P; // true
```

What to Submit

Define the class SquareMatrix in a file SquareMatrix.h and provide its implementation in SquareMatrix.cpp. Include main() in the file main.cpp for testing. Submit these to the Moodle site by the deadline.