Submission

- Matrix.h, Matrix.cpp, RowItr.h RowItr.cpp, ColItr.h, ColItr.cpp, RowElementItr.h, RowElementItr.cpp, ColElementItr.h, ColElementItr.cpp, MatrixError.h, MatrixError.cpp, MatrixSizeError.h, MatrixSizeError.cpp, MatrixSizeMismatchError.h, MatrixSizeMismatchError.cpp, MatrixInnerDimensionMismatchError.h, MatrixInnerDimensionMismatchError.cpp
- Time it took Matthew: 2 hours

Description

For this problem you will be implementing a matrix of **doubles** that has scalar addition, scalar multiplication, matrix addition, matrix multiplication, outstream operators, and instream operators defined on it as well as row-wise and column-wise iterators defined on it. The functions that you must implement are defined in the given .h files. You can add more functions to these files if you wish.

Scalar Addition

A scalar is a single number like 3.5, 12, or -39. To add a scalar and a matrix together you add the scalar to every element in the matrix. For example let the matrix M =

1	2	3
4	5	6

Then M + 10 is

11	12	12
14	15	16

Scalar Multiplication

Scalar multiplication is very similar to matrix addition but instead of adding the scalar to each element in the matrix you multiply it by each element. For example if M =

1	2	3
4	5	6

Then M * 10 is

10	20	30
40	50	60

Matrix Addition

A description of how matrix addition works can be found here: http://www.purplemath.com/modules/mtrxadd.htm . If the size of the matrices are not the same size your function should throw a MatrixSizeMismatchError who's error string reads: `"Matrices must be the same size but Matrix1 is A X B but Matrix2 is C X D."` Where A is the number of rows in the first matrix, B the number of columns in the first matrix, C the number of rows in the second matrix, and D the the number of columns in the second matrix.

Matrix Multiplication

A description of how matrix multiplication works can be found here: http://www.purplemath.com/modules/mtrxmult.htm . If the inner dimension of the matrices, the number of columns in A and the number of rows in B in the expression A* B are not the same you should throw a MatrixInnerDimensionMismatchError who's error string reads: "Inner dimensions do not agree. First matrix has B columns but second matrix has C rows." Were B is the number of columns in the first matrix and C is the number of rows in the second matrix

Hints

- Don't forget that A*B != B*A. Order matters in matrix multiplication. It isn't commutative

- Don’t forget in your *= operator that doing matrix multiplication is going to change the number of columns.

Outstream and Instream Operators

Your matrix class should overload the outstream << and instream operators >>. For the outstream operator you should print out the elements of the matrix row by row. For the instream operator the input will be

- NumRows NumCols
- Matrix values

Iterators

Your matrix class should have rowBegin/colBegin and rowEnd/colEnd methods that return Row/Col Iterators to the beginning/end of the matrix. The RowItr iterates over the rows of the matrix and the ColumnItr over the columns of the matrix. When dereferenced, the RowItr should return a RowElementItr which iterates over the elements in that row of the matrix. When dereferenced, the ColItr should return a ColElementItr which iterates over the elements in that column of the matrix. Somewhat unique to the the RowItr and ColItr is that they implement operator[]. They should return a RowElementItr/ColElementItr over the row that is specified.

Exceptions

You will be defining 4 Exceptions for this program

1. Matrix Error: the base class for the matrix errors. It should inherit from std::exception
2. MatrixSizeError: to be raised if the number of elements in each row aren’t the same
3. MatrixSizeMismatchError: to be raised if the operation requires both matrices to be of the same size but they are not (matrix addition)
4. MatrixInnerDimensionMismatchError: to be raised if the operation requires the inner dimensions to agree (matrix multiplication)

In addition to these user defined operators your iterators will raise std::out_of_range exceptions if they are ever dereferenced out of bounds. There error message should look like: “IteratorType dereferenced out of bounds. Acceptable values 0 - UpperBound. Current position: curPos” Where IteratorType is the type of the iterator (RowItr, ColItr, RowElementItr, ColElementItr), UpperBound is the maximum range of the iterator and curPos is the index of the value it is currently referring to.

Hints

- Start with instream and outstream operators

- Then do the scalar operators
- Then do matrix addition
- Then matrix multiplication
- Then do the iterators (this is the most time consuming part)
    - You may be tempted to use inheritance here as much of the code is the same but it won't quite right due to the different return types. You'd need both inheritance and templates to do it this way so save yourself some time and copy and paste for now.
- For an extra challenge and practice try making it so that you can do range based for loops on your matrix