MQF Object Oriented Programming I Fall 2019
Hw4 Due 11/20/2019 before midnight

Specifications

Create a class call it '***Matrix'***. A user can use this class to perform matrix operations such as matrix addition, subtraction, and multiplication, assign one matrix to the other matrix, output a matrix, input a matrix, etc.

Requirements

You have to write a **class** call ***Matrix*** which contains required member functions, setters, getters, and helpers, and most importantly overload operators such as +, -, *, =. Non-member friend functions are also needed for matrix input (overload >>) and output (overload <<). In addition to these, this Matrix class also have one function to return an *identity* matrix of same dimensions (of course must be square matrix). Another function is *transpose* that returns a ***Matrix*** object which will turning all rows of this matrix into columns and vice versa. Note *n by m* matrix will return an *m by n* matrix.

Write a main program **matrix-test**.cpp which will use and test ***Matrix***.cpp. If you use dynamic memory allocation to implement your class, you need to handle memory-leak issues. Make sure that you write constructors, destructor, and overload assignment operator =. Assume in this assignment, a constructor with no argument will create a matrix of *3 x 3* with all entries *0*. A constructor with one argument, say *n*, will create an *n x n* matrix with all entry values *0*. A constructor with two arguments, say *n* and *m*, will create an *n x m* matrix with all entries *0*. A constructor with three arguments, say *n*, *m*, and *v*, will create an *n x m* matrix with all values *v*. There is also a copy constructor which will take a matrix as its argument.

Implementation of the ***Matrix*** class: although you may assume all entries of the matrix are integers, using double make your class more useful. For example, you may extend your class to add a function *inverse* to compute an inverse of this matrix if this matrix is square and not singular.

Testing

Come up with your own testing plan. You shall test matrix addition, subtraction, multiplication, transpose, input, and output. Output error message if operation is not possible. For example dimensions of Matrices are not matching so cannot do addition or subtraction, etc.

Grading rubrics

(1) Programming style	5
(2) Use operator overloading	5
(3) Use of class	5
(4) Appropriate testing	5
(5) Correct results	15

Extra credit (5 points)

Add an inverse function which will return an inverse matrix if this matrix is square and not singular.

Submission:

(1) Run your program and record the output by copy the screen shots
(2) Put all your screen shots into a document and convert it into a pdf file
(3) Put all your source code file or files (.cpp file, .h) into a folder. Use *yourLastName-FirstName-Hw4* as the name of the folder. E.g. if your name is Kathy Smith then *Smith-Kathy-hw4* is the folder name
(4) Put the test run screen shot files (as thorough as possible) into the folder above
(5) Zip the folder and Use *yourLastName-FirstName-Hw4* as the zip file name.
(6) Submit the file to blackboard.