Assignment 7

CAAM 519 Fall 2021

due Monday December 13, 11:59 pm

In this assignment, you will build a templated matrix class. You can start from the files placed in the website and complete the codes. You are encouraged to collaborate and discuss solutions, but your code and comments must be entirely your own. (Total: 50 pts)

- (20pts) Create a header file called `my_matrix.h`. Within this header file you will define and implement a templated matrix class with the following members andz functions:
  - Private member integers corresponding to the number of rows and number of columns.
  - A private member string that corresponds to the matrix name. Be sure to include the header file `string` so you can define something of type `std::string`.
  - A private member of type `T**` that contains elements of type `T` contained in the matrix.
  - A default constructor that sets the number of rows and columns to 2 and the name to "default." Please use c`++` style memory management in all cases with the operators `new` and `delete`, and allocate memory *contiguously*, in column major format. Contiguous memory will be important for when you pass your matrices into Fortran functions.
  - A user constructor which takes as input the number of rows, number of columns, and the matrix name.
  - A copy constructor.
  - A destructor.
  - A public function called `display` that prints out the matrix name and its elements to the terminal. For example, the 2×2 identity matrix with name "A" would look like:

    ```
    A =
    1 0
    0 1
    ```

    when displayed in the terminal.
  - An overloaded assignment operator "=."
  - An overloaded plus operator "+."
  - An overloaded element access operator "()" so that `A(ii,jj)` should return a writeable reference to the element of the matrix `A` in the `ii`th row and `jj`th column.
  - An overloaded left division operator "|" that computes $A^{-1}b$.

  This class should be templated on the type `T` of the matrix elements. Note that since this is a templated class, I would recommend implementing the methods and operators above in the same header file `my_matrix.h`, but outside of the class definition. The left division operator will be implemented by calling the appropriate LAPACK function and using template specialization. See the third point below.

- (10pts) Some of the member functions should throw exceptions in certain cases, given below. You should throw a `char*` that contains information about the error and also the function in which it occurs.

  - In the user constructor if the number of rows or columns are less than or equal to zero.
  - In the operator "=" if the input matrix has different dimensions from the current matrix.
  - In the operator "()" if the input dimensions are out-of-bounds for the current matrix.
  - In the operator "+" if the two matrices you are summing have different dimensions.

- (5pts) For the left division operator, we will call a LAPACK function. Since our matrix class is templated, we need to specialize the implemention of this operator.
  - Create a header file called `lapack_prototypes.h` that contains the prototypes for `dgesv` and `sgesv`, used for doubles and floats respectively. Note the rules for name mangling Fortran and C functions that we discussed in class.
  - Implement the left division operator for the cases when your template parameter is either a `double` or a `float`. Note that you should put these implementations into your `my_matrix.h` header file.

- (10pts) Now time to test out your matrix class! When you populate your matrices, use your overloaded "()" operator, and when you want to print out your matrices to the terminal, please call the `display` function you have implemented. The following tasks should be put sequentially in a source file called `main.cpp`.
  - Construct a 2×2 `float` matrix called `A1` with name "A1." Populate it with the following values:

$$A_1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$

    Create a matrix called `A2` using the copy constructor with `A1`. Create a matrix `A3` with the default constructor, and use the plus operator to sum `A1` and `A2` and set the result equal to the matrix `A3`. Print out `A3` to the terminal.
  - Construct a 3×3 `double` matrix called `A4` with name "A4." Populate it with the following values:

$$A_4 = \begin{bmatrix} 1 & 3 & 4 \\ -1 & 5 & 1 \\ 0 & 2 & 1 \end{bmatrix}.$$

    Construct a 3×1 `double` matrix called `B` with the name "B." Populate it with the following values:

$$B = \begin{bmatrix} 21 \\ 21 \\ 10 \end{bmatrix}.$$

    Use the overloaded left division operator to solve the linear system $A_4 x = B$. Print out A_4, B, and the solution x to the terminal.

- (5pts) Write a Makefile to automate and optimize the compilation process.

Make sure your code is free of memory leaks. The graders will run the code through valgrind. Please upload your source, header files, and Makefile to Canvas.