CAAM 420 Fall 2012 Homework 5

T. Warburton

October 24, 2012

You must complete the following task by 5pm on Monday 11/05/12.

Your solutions to the homework must be committed to your Subversion repository in a sub-directory HW05.

You will commit your files as follows:

i. Use \LaTeX\ and Kile to write and format your homework.

ii. Submit the following files HW05/hw05.pdf, HW05/hw05.tex.

iii. your source code files in HW05/hw05.cpp, HW05/dmatrix.h

iv. your Makefile in HW05/Makefile

Work individually on this task. Do not work with other students in the class. You may consult the internet and programming texts.
Q1 Purpose: practice building C++ objects, in particular constructors, destructors, member functions, operators, friend functions, and optionally output to streams. Calling Fortran from C++. Each component is straight forward and exercises some aspect of programming in C++ discussed in class.

In this homework you will create a relatively simple C++ matrix class. The homework tasks are:

- [10 points] **class dmatrix**
  - Start a header file `HW05/dmatrix.h` that includes the definition of a `dmatrix` class with appropriate private member variables and definitions and implementations for the following member functions and operators.

- [5 points] **default constructor**
  - Creates a `dmatrix` class object with zero dimensions.

- [5 points] **constructor**
  - Initializes a `dmatrix` using user supplied number of rows and number of columns.

- [10 points] **copy constructor**
  - Initializes a `dmatrix` using the dimensions and data from a second `dmatrix`.

- [5 points] **destructor**
  - Deletes memory dynamically allocated for the `dmatrix`.

- [10 points] **randomize**
  - Member function that sets the entries of the `dmatrix` to be random numbers in the range [0, 1], using the `drand48` stldlib random number generator.

- [10 points] **operator ()**
  - Overloaded bracket operator() to perform combined role of get and set function. This will return a reference to an entry in the `dmatrix` at a requested row and column.

- [10 points] **operator +**
  - Overloaded binary operator+ friend of the `dmatrix` class to perform matrix addition with two `dmatrix` objects.

- [20 points] **operator ***
  - Overloaded binary operator* friend of the `dmatrix` class to perform matrix multiplication with two `dmatrix` objects using the BLAS `dgemm` subroutine.

- [25 points] **operator |**
  - Overloaded binary operator | friend of the `dmatrix` class to perform left division using the LAPACK `dgesv` subroutine.

- [10 points] **print**
  - Member function that prints the `dmatrix` with C style IO.

- [10 points] In your \LaTeX document create a detailed subsection for each function described above. Use the \LaTeXXmath environments and commands to describe the matrix-matrix and matrix inversion performed in parts viii and ix. You do not need to give detail of how `dgesv` LAPACK function performs the matrix inversion.

- [10 points] Use your `dmatrix` class to complete the following main function (download):

Listing 1: main function

```cpp
#include "dmatrix.h"

main () {

    // constructor
dmatrix A(4,4, "A");
dmatrix B(4,4, "B");

    // randomize entries
    A.randomize();
}
```
B. randomize();

// overloaded binary operator* for matrix–matrix multiplication
dmatrix C = A * B;

// overloaded binary operator| for left matrix division
dmatrix D = A | C; // i.e. D = A\C in MATLAB
dmatrix E = B | D;

// complicated expression with simple outcome
dmatrix F = (A*B) | ( (A+B) | ( (B+A)*A + (A+B)*B ) ) ;

// test the overloaded two-argument operator()
F(4,1) = 2.;

A. print();
B. print();
C. print();
D. print();
E. print();
F. print();
}

xiii. [20 points] Include the output from running this main function in your report using the \verbatim environment. Explain any deviations from expected output.

xiv. [20 points] Include your C++ listings in the \LaTeX report using the \lstlisting environment provided in the listings package.

xv. [Extra credit: 20 points] Add a string to your dmatrix that encapsulates in text how the dmatrix was formed. Examine the string for matrix F to explain what sequence of functions and operators were called in its creation. You will need to add extra code to each constructor and overloaded operator to modify the string appropriately.