You must complete the following task by 5pm on Monday 11/17/14.

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

You will commit your files as follows:

1. Use \LaTeX and Kile to write and format your homework.
2. Submit the following files HW05/hw05.pdf, HW05/hw05.tex.
3. Your source code files in HW05/main.cpp, HW05/matrix.hpp
4. 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, templating and output to a file stream Calling Fortran from C++

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

o. [10 points] class matrix
   Start a header file HW05/matrix.hpp that includes the definition of a templated matrix class with appropriate private member variables and definitions and implementations for the following member functions and operators. The templating class type will specify whether the matrix has float or double type.

i. [5 points] default constructor
   Creates a matrix class object with zero dimensions.

ii. [5 points] constructor
    Initializes a matrix class object using user supplied number of rows and number of columns.

iii. [10 points] copy constructor
    Initializes a matrix using the dimensions and data from a second matrix.

iv. [5 points] destructor
    Deletes memory dynamically allocated for the matrix.

v. [10 points] randomize
    Member function that sets the entries of the matrix to be random numbers in the range [0,1], using the drand48 stdlib random number generator.

vi. [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 matrix at a requested row and column.

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

viii. [20 points] operator *
     Overloaded binary operator* friend of the matrix class to perform matrix multiplication with two matrix objects using the BLAS sgemm or dgemm subroutines depending on the templating type.

ix. [25 points] operator |
    Overloaded binary operator | friend of the matrix class to perform left division using the LAPACK sgesv or dgesv subroutine depending on whether the matrix is single or double precision (Hint: use template specialization).

x. [10 points] operator <<
    Overloaded stream out operator that streams the matrix into a user specified output stream.

xi. [10 points] In your LaTeX document create a detailed subsection for each function described above. Use the LaTeXmath 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 the sgesv and dgesv LAPACK functions perform the matrix inversion.

xii. [10 points] Use your matrix class to complete the following main function (download):

Listing 1: main function

```c++
#include <iostream>
#include <fstream>
using namespace std;

#include "matrix.hpp"

main()
{
  // constructor
```
matrix<float> A(4,4, "A");
matrix<float> B(4,4, "B");

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

// overloaded operator* for matrix-matrix multiplication
matrix<float> C = A * B;

// overloaded operator/ for left matrix division
matrix<float> D = A / C; // i.e. D = A/C in MATLAB
matrix<float> E = B / D;

// complicated expression with simple outcome
matrix<float> F = (A + B) / ( (A + B) | ( (B + A)*A + (A + B)*B ) );

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

// stream out to standard out
cout << A << endl;
cout << B << endl;
cout << C << endl;
cout << D << endl;
cout << E << endl;
cout << F << endl;
}

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++ source code in the \LaTeX\ report using the \lstlisting environment provided in the listings package. [HINT: see main.tex from EX16].

xv. [10 points] Provide a clean valgrind report for your code.

xvi. [20 points] Add a C++ string to your matrix that encapsulates in text how the matrix was formed. Examine the string for 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.

xvii. [Extra credit: 40 points] Add single precision and complex precision template specialization for the matrix class. You will need to use \texttt{cgemm, cgesv} for single precision complex and \texttt{zgemm, zgesv} for double precision complex matrix operations. Hint: Fortran stores complex numbers in 8 bytes (single precision) and 16 bytes (double precision) using the convention that the first half of each complex variable is the real part and the second part is the imaginary part. You should test using complex valued matrices. Finally, you will need to create a C++ struct or class type for complex and doublecomplex for the templating variable type as they are not intrinsic types in C++.