HBM803E - Object Oriented Programming Techniques

HOMEWORK I (due to Friday 04/12/2020 17:00PM)

Questions

Q1 (100pt)

1. **(10pt)** Create a class named sqMatrix, in order to store and manipulate 2 dimensional square matrices, that have following attributes:
   - integer N – length of the matrix in both directions (only one dimension will be stored, because it is square)
   - a two dimensional array of type "double" that stores matrix

2. **(20pt)** The sqMatrix class should have
   a non-default constructor that takes an integer number as argument, initialize objects argument N as this integer number, allocate memory for 2d N*N double type array and initialize it with zeros (test it with a code line in main function, e.g. a line similar to "sqMatrix A(5);") **[5pt**],

   a deconstructor that deallocate memory allocated by constructor (test it with a code line in main function, e.g. a line similar to "delete[] A;") **[5pt],**

   a get() method that gets elements of matrix from user input or screen, and a show() method that print out the content of the matrix in a following form (test them in the main function by using get() first and screening same elements by using show()) **[5pt]**

```
*** init ***
     0       1       2
0   0.00    0.00    0.00
1   0.00    0.00    0.00
2   0.00    0.00    0.00

*** filled ***
     0       1       2
0   3.00   -0.10   -0.20
1   0.10    7.00   -0.30
2   0.30   -0.20   10.00
```

   a transpose() method to manipulate matrix elements to make it transpose of the original matrix **[5pt]**

3. **(15pt)** Overload =, +, -, += and -= operators to handle corresponding operations (such as A=B, A=3.5 by making every element of A equal to 3.5, A+= B, A += 4.1 by adding 4.1 to every element of A, C=(A+B) , C=(A+3.2) by making all elements of C 3.2 bigger than corresponding element of A without changing elements of A; here A, B and C are objects of sqMatrix class with same matrix size N) **[5pt]**;

when used inappropriately(with matrices of different size), they should return an error as return value, keep attributes of used matrices same, and print an error message on the screen **[5pt]**,

a friend function (a friend overloaded operator) that handles 3.5 + B operation by adding 3.5 to every element of B (should not change attributes of B) and return result (test it via an operation like C=5.1 + B) **[5pt]**

4. **(45pt)** A class method for gauss elimination linear equation system solving method named GaussianElimination:
- this method should take a one dimensional array "Y" of type double and of length N as argument, and return an other one dimensional array "X" of type double and of length N (creating a new array as "X" is optional as long as elements of input array Y are preserved – not changed) **[5pt]**
- implement Gauss Elimination method and solve for Y **[35pt]**
- it should preserve (not change) original matrix values **[5pt]**

Hints:

Transpose of a matrix is a matrix (i,j)th element is equal to (j,i)th element of the original matrix.

Gaussian elimination, also known as row reduction, is an algorithm in linear algebra for solving a system of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients.

Gaussian Elimination pseudo-code for a square matrix:

```
// A(i,j) means element of matrix A in row i and column j.
// A(i,:) means ith row of matrix A and A(:,i) is ith column
// y(i) means ith element of array y
// Pivot and Factor are variable names


i:0 to n-1
// partial pivoting
        set Pivot to row number of the greatest absolute value in column i
( A(:,i) )
        swap row i with row Pivot of matrix AND element i with element
Pivot of Y


// getting reduced row-echelon form
      A(i,:) := A(i,:) / A(i,i)
```

```
y(i) := y(i) / A(i,i)
j:0 to n-1, except i
        A(j,:) := A(j,:) - A(i,:)
        y(j) := y(j) - y(i)
```

In this pseudo code, method applies appropriate row operations to the matrix A in order to transform it into the reduced row echelon form which is equal to identity matrix I in case of square matrix, while applying the same row operations to column vector Y. Resulting Y becomes the solution for x in the linear equation system described by Ax=y.

An alternative implementation can get the row-echelon form and use back substitution for solution.

To increase numerical stability of the method, largest possible absolute value in the column should be chosen as pivot. (called partial pivoting)

5 bonus points will be provided (if total of earned points is equal or less than 95) if class definition is written in a header file, and header file is included in answer codes for questions

Rules

- As with all programs in this course, your code should contain useful comments. In particular, your name, the date, and a brief description of what the program does should appear at the top of your source file.
- Each question should have its own folder, which contains source code and Makefile to compile it. The homework can be given as an Eclipse project but in this case each question will be defined as different project.
- All files must be packed and compressed as a single file with following naming convention,
  hw-01-[STUDENT-ID].tar.gz
  The file must be uploaded to course Moodle site before its deadline:
  http://ninova.itu.edu.tr/

Hint

Following Linux commands can be used to create tar file

- tar -cvf hw-01-[STUDENT-ID].tar [DIRECTORY-TO-TAR]
- gzip hw-01-[STUDENT-ID].tar