DALHOUSIE UNIVERSITY
DEPARTMENT OF ENGINEERING MATHEMATICS
ENGM3282

ASSIGNMENT # 10, Due date: Thursday November 29, 2018, 1:00 PM

The header file `matrix.h` declares a matrix class and various matrix functions.

The source file `matrix.cpp` implements most of the methods and functions declared in `matrix.h`.

In each question of this assignment you will implement one of the remaining functions and test your implementation using a test program.

1. Implement the function `solve(matrix a, matrix b)` which solves the matrix equation $a*x = b$ for x using Gauss–Jordan Elimination with partial pivoting on the augmented matrix $[a \;\; b]$ and returns the solution.

   Note that `a` and `b` are passed by value as the algorithm modifies both matrices turning `a` into the identity matrix and `b` into the solution.

   Use the following steps:

   - if $a.rows()! = b.rows()$ then print the message that the matrices are not compatible and exit.
   - if $a.rows()! = a.cols()$ then print the message that the coefficient matrix is not square and exit.
   - Let $n = a.rows()$ and use the for loop:

   ```
   for(int p = 0; p < n; p++) {

       find the row, k, with k >= p where |a(k,p)| is largest

       if |a(k,p)| < matrix::tiny then print the message that the
           coefficent matrix is not invertible and exit

       interchange rows k and p of the matrix a
       interchange rows k and p of the matrix b

       divide through row p of a by a(p,p) (makes a(p,p) = 1)
       divide through row p of b by a(p,p)

       for all rows i != p, eliminate a(i,p) by subtracting
           a(i,p) * (row p of a) from (row i of a) (makes a(i,p) = 0)
           also subtract a(i,p) * (row p of b) from (row i of b)
    }
   ```

   - return b

   Add your source code for the `solve()` function to matrix.cpp and use the following test program to test your `solve` function.

```
/*  File: testsolve.cpp
    test the solve function for the matrix class */

#include "matrix.h"

int main(void)
```

```
{
    ifstream fin("testsolvein.txt");

    matrix a(2,2), b(2,4), x;

    fin >> a >> b;

    cout << "a = " << endl;
    cout << a << endl << endl;

    cout << "b = " << endl;
    cout << b << endl << endl;

    x = solve(a,b);

    cout << "x = " << endl;
    cout << x << endl << endl;

    cout << "check a * x should be b" << endl;
    cout << a * x << endl << endl;

    fin.close();
    return 0;
}
```

Use the input file `testsolvein.txt` consisting of

```
1  2
2  3

5  4  1  2
2  1  3  4
```

To compile and run your test program you must make a project consisting of the two source files `matrix.cpp` and `testsolve.cpp`.

For marking purposes submit your source code for `solve` along with your program output.

2. Implement the function `matrix inverse(const matrix &a)` which returns the inverse of `a` if it exists.

   Use the following steps:

   - Let $n = a.rows()$ and let $b = eye(n)$ (the n x n identity)
   - solve the matrix equation $a * x = b$
   - return x (will be the inverse of a)

   Add your source code for the `inverse()` function to matrix.cpp and use the following test program to test your `inverse` function.

```
/*  File: testinverse.cpp
    test the inverse function for the matrix class */

#include "matrix.h"
```

```
int main(void)
{
    ifstream fin("testinversein.txt");

    matrix a(2,2), x;

    fin >> a;

    cout << "a = " << endl;
    cout << a << endl << endl;

    x = inverse(a);

    cout << "inverse = " << endl;
    cout << x << endl << endl;

    cout << "check a * x should be the identity matrix" << endl;
    cout << a * x << endl << endl;

    fin.close();
    return 0;
}
```

Use the input file `testinversein.txt` consisting of

```
1  2
2  3
```

To compile and run your test program you must make a project consisting of the two source files `matrix.cpp` and `testinverse.cpp`.

For marking purposes submit your source code for `inverse` along with your program output.

3. A matrix equation $a * x = b$ where the matrix a is m x n is called **overdetermined** if $m > n$. That is, there are more equations than unknowns.

   We usually cannot solve such systems but we can find an approximate solution x such that $a * x$ is as close to b as possible.

   The distance between two matrices is the euclidean distance between the matrices as if they are vectors (square root of the sum of squares). Hence the approximate solution, x, is called the **least squares solution**.

   The least squares solution satisfies $a^t * a * x = a^t * b$, where a^t is the transpose of a.

   Implement the function `matrix leastsquares(const matrix &a, const matrix &b)` which returns the least squares solution of $a * x = b$.

   Use the following steps:

   - Form the matrices $a_1 = transpose(a) * a$ and $b_1 = transpose(a) * b$
   - solve the matrix equation $a_1 * x = b_1$
   - return x

   Add your source code for the `leastsquares()` function to matrix.cpp and use the following test program to test your function.

```
/*  File: testleastsquares.cpp
    test the least squares function for the matrix class */

#include "matrix.h"

int main(void)
{
    ifstream fin("testleastsquaresin.txt");

    matrix a(3,2), b(3,1), x;

    fin >> a >> b;

    cout << "a = " << endl;
    cout << a << endl << endl;
    cout << "b = " << endl;
    cout << b << endl << endl;

    x = leastsquares(a,b);

    cout << "x = " << endl;
    cout << x << endl << endl;

    cout << "check a * x should be as close to b as possible" << endl;
    cout << a * x << endl << endl;

    fin.close();
    return 0;
}
```

Use the input file `testinversein.txt` consisting of

```
1  2
2  3
1  1

5
7
1
```

To compile and run your test program you must make a project consisting of the two source files `matrix.cpp` and `testleastsquares.cpp`.

For marking purposes submit your source code for `leastsquares` along with your program output.

4. The least squares solution to $a * x = b$ satisfies $a^t * a * x = a^t * b$, where a^t is the transpose of a.

   Using the inverse of $a^t * a$ we can write the solution as $x = (a^t * a)^{-1} * a^t * b$.

   The matrix $p = (a^t * a)^{-1} * a^t$ is called the pseudoinverse of a.

   Note that if a is m by n then the matrix p is n by m and satisfies $p * a = I_n =$ the n by n identity matrix.

   Implement the function `matrix pseudoinverse(const matrix &a)` which returns the pseudoinverse of a

   Use the following steps:

- Form the matrix $a_1 = transpose(a) * a$
- Form the inverse $a_2 = inverse(a_1)$;
- Form $p = a_2 * transpose(a)$
- return p

Add your source code for the `pseudoinverse()` function to matrix.cpp and use the following test program to test your function.

```
/*  File: testpseudoinverse.cpp
    test the pseudoinversefunction for the matrix class */

#include "matrix.h"

int main(void)
{
    ifstream fin("testpseudoinversein.txt");

    matrix a(3,2), x;

    fin >> a;

    cout << "a = " << endl;
    cout << a << endl << endl;

    x = pseudoinverse(a);

    cout << "x = " << endl;
    cout << x << endl << endl;

    cout << "check x * a should be the identity matrix" << endl;
    cout << x * a << endl << endl;

    fin.close();
    return 0;
}
```

Use the input file `testpseudoinversein.txt` consisting of

```
1  2
2  3
1  1
```

To compile and run your test program you must make a project consisting of the two source files `matrix.cpp` and `testpseudoinverse.cpp`.

For marking purposes submit your source code for `pseudoinverse` along with your program output.