Homework 5 (20 points)

Objects

March 3, 2026

1 HW5 contents

This file comes zipped with three files:

1. Simple object library matrix.cpp (optimized for simplicity, not speed).
2. Header file matrix.h for the matrix library (contains mostly declarations of the class and functions).
3. Demo program mainMatrix.h, which uses the matrix library.

Unzip all three into the same directory. Create a new project in your IDE, add the $*$.h and both $*$.cpp files to the project. Rebuild and run.

2 Your tasks

1. (7 points total) Read through matrix.h, matrix.cpp and the demo program, and understand it. Answer the following questions:
   (a) (2 points) What does the assignment (i.e. = 0) in the declaration of the first constructor mean?
   (b) (2 points) What is the pointer **this**? Why is it used in one of the constructors? Can it be used in the second constructor, and how?
   (c) (3 points) What is a copy constructor and why is it necessary here? Hint:

```
matrix a(5,5), b = a;
cout <<"a == " <<endl <<a <<endl;
cout <<"b == " <<endl <<b <<endl;
cout <<"Randomizing a ... Now" <<endl;
a.randomize();
cout <<"a == " <<endl <<a <<endl;
cout <<"b == " <<endl <<b;
```

   Check what happens with and without the copy constructor (do not forget to rebuild the whole project before running).

2. (4 points total) Add the following new features to the class matrix and write your own short demo program (modify mainMatrix.cpp) to demonstrate that your overloaded operators do really work.

   (a) (2 points) Add the following declarations to the class definition

   ```
   matrix& operator-= (const matrix& m);
   matrix& operator/= (const double& v);
   ```

   Write the corresponding definitions in matrix.cpp (should be quite similar to **operator**+= and **operator**∗=).

   (b) (2 points) Add the following declarations of global functions to matrix.h:

   ```
   matrix operator- (const matrix & m1, const matrix & m2);
   matrix operator/ (const matrix & m, const double & v);
   ```

   Write the corresponding definitions in matrix.cpp (should be quite similar to **operator**+ and **operator**∗).

3. (4 points) Assume the member function matrix::transpose() is defined (in matrix.cpp) as follows:

   ```
   matrix matrix::transpose() {
      matrix tmp(colNo,rowNo);
      for(unsigned int r=0; r<=rowNo-1; ++r)
         for(unsigned int c=0; c<=colNo-1; ++c)
            tmp(c,r) = (*vals)[r][c];
      (*this) = tmp;  // operator= of the class matrix
      return *this;
   }
   ```

   and accordingly declared in matrix.h, i.e. someone forgot the reference operator & and so the transposed matrix is returned by value and not by reference. What would be the effect of executing the following code:

   ```
   matrix a(5,5), b(5,5);
   a.randomize();
   cout <<"a == " <<endl <<a <<endl;
   cout <<"b == " <<endl <<b <<endl;
   cout <<"Performing b = a.transpose().transpose();" <<endl;
   b = a.transpose().transpose();
   cout <<"a == " <<endl <<a <<endl;
   cout <<"b == " <<endl <<b <<endl;
   ```

   with the original and the modified function? Why the difference?

4. (5 points) Write (i.e., properly declare and define) a new member function trace() of the class matrix, which would return the trace of the matrix (sum of the elements on the diagonal).

E-mail your answers and reworked source codes (all three files) to `ljank@ippt.pan.pl`.