Exercise 09:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, December 3rd (16:30).

Question 1: Operator overloading (100 points).

In this question you will exercise your understanding of how operator overloading can be used. In this exercise, your goal is to define a class Mat3x3 that represents 3 x 3 matrices over R (use double to store the coefficients). The default constructor of this class creates I3, the identity matrix. Another constructor of this class takes as argument an array of array (double[][]) containing the matrix coefficients (the element at position [i][j] in this array corresponds to the (i,j) element of the matrix). Define the following operations on Mat3x3:

- '+', '-': addition and subtraction of two matrices. Example: C = A + B where A, B and C are Mat3x3 objects
- '*': matrix multiplication. Example: C = A * B where A, B, C are Mat3x3 objects
- '+=', '-=', '*=': compound addition / subtraction / multiplication and assignment operator: B += A is equivalent to B = B + A, where A and B are Mat3x3 objects
- '-': unary minus. Example: B = -A where A and B are Mat3x3 objects; B's coefficients correspond to -A's coefficients.

- '(i,j)': used to access the i,j (ith row, jth column) element of the matrix. Example: double aij = A(i, j); where A is a Mat3x3 object
- '==': to test if two Mat3x3 matrices are equal. Two matrices are equal if their coefficients are equal. Example: if (A == B), where A and B are Mat3x3 objects
- '<<': the stream insertion operator, that allows to insert Mat3x3 objects in C++ streams

Use "Mat3x3.h" and "Mat3x3.cpp" for the filenames. You can use any internal representation that you want for storing the matrix coefficients within the class Mat3x3. To test your code, you can type the following code in a file "test_Mat3x3.cpp":

```
#include <iostream>
#include <cassert>
#include "Mat3x3.h"

using namespace std;

int main(void) {
 Mat3x3 A; // A = I3
 double coefficients[3][3] = {{1.0,2.0,3.0},{4.0,5.0,6.0},{7.0,8.0,9.0}};
 Mat3x3 B(coefficients);

 Mat3x3 C = A * B;
 assert(C == B);

 C += B;
 C -= A;
 assert(C(0,0) == 1.0);
```

```
 assert(C(1,1) == 9.0);
 assert(C(2,2) == 17.0);

 Mat3x3 D = -A;
 assert(D(0,0) == -1.0);
 assert(D(1,1) == -1.0);
 assert(D(2,2) == -1.0);

 cout << "Tests passed" << endl;

 return 0;
}
```