**HW4: Overloaded operators.**

Create class Matrix,

which has ***member variable* e:** nxn array of integers (const int n=3),

***member functions:***

* Matrix() //default constructor creating a matrix with elements =0,
* Matrix(int d) //constructorcreating a diagonal matrix (d-s on the main diagonal, 0 –everywhere else)
* int det() //function computing determinant of a matrix <http://mathworld.wolfram.com/Determinant.html> formula (10)

<http://www.mathworks.com/help/aeroblks/determinantof3x3matrix.html> (in this formula indices of matrix elements go from 1 to 3, while indices of array go from 0 to 2);

and **overloaded operators:** <<, >>, ==, **+, -** (binary and unary), **\*,** defined as follows:

* << and >> are operators for file/console I/O
* A==B if for all i,j: A.e[i][j]==B.e[i][j]
* B=-A means that for all i,j: B.e[i][j]=-A.e[i][j];
* C=A+B means for all i,j: C.e[i][j]= A.e[i][j]+B.e[i][j];
* C=A-B means for all i,j: C.e[i][j]= A.e[i][j]-B.e[i][j];
* C=A\*B means for all i,j: C.e[i][j]= i-th row of A \* j-th column of B=sum for k of (A.e[i][k]\*B.e[k][j]) for k from 0 to 2.

In the program test class Matrix, its functions, and operators:

* create Z – matrix with all zeroes (using default constructor), output Z to the screen;
* create E - diagonal matrix with **1** on the main diagonal, 0 – everywhere else, output E;
* create D - diagonal matrix with **2** on the main diagonal, 0 – everywhere else, output D;
* create A with elements inputted from file, output A;
* create B as a copy of A; check that B== A (using == operator);
* compute and **output** to the screen the following matrices (displaying what is being done)
  + A+D //cout <<”A+D==”<<A+D<<endl;
  + A-D
  + A\*D
* check (with ==) that
  + A-B==Z,
  + –A==Z-A,
  + A+B == A\*D,
  + A\*E==A
  + A\*D==2\*A
* compute determinants of E and D (should = 1 and 8)
* create C with elements inputted from file;
* check the following property of the determinant: (A\*C).det()==A.det()\*C.det().

**Submit input file, code, and output.**