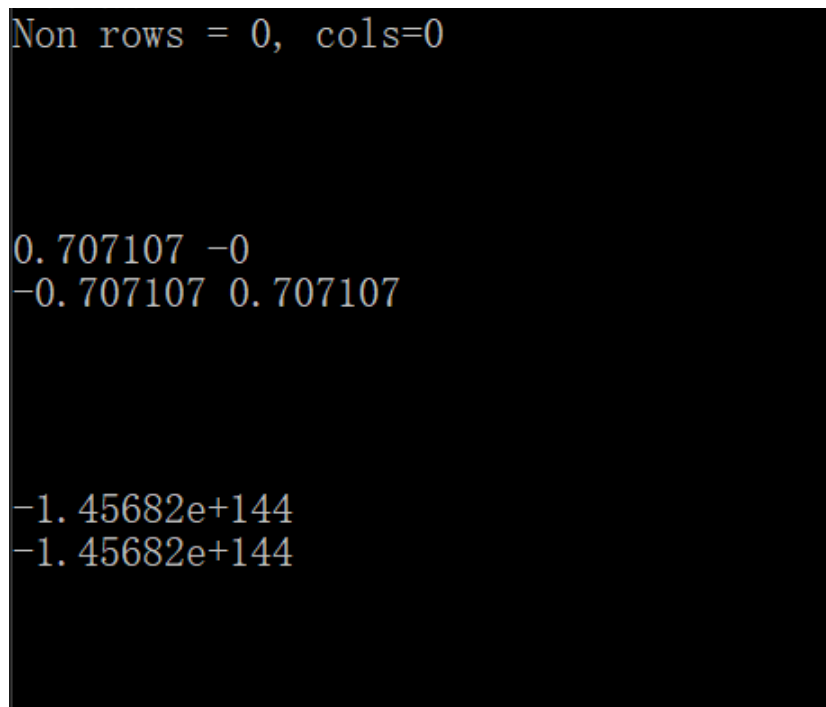


1. Experimental tasks and requirements:

- 1) Write a class Matrix that represents a matrix. Matrix elements are of double type.
- 2) Rewrite **the default constructor** to generate an empty matrix (no memory allocation, the matrix size is 0*0).
- 3) Add a **constructor with parameters**, the input parameter is the size of the matrix, the structure allocates matrix memory, and initializes the matrix elements to 0. Matrix elements are stored in row order.
- 4) Free the memory in **the destructor**.
- 5) Write **a copy constructor** and **a copy assignment function** (operator =). (note the use of deep copy)
- 6) Write the member function Set Value to realize the assignment of row i and column j of the matrix.
- 7) Write the member function GetValue to return the value of row i and column j of the matrix.
- 8) Write the member functions GetRowCount and GetColCount to obtain the number of rows and columns of the matrix respectively.
- 9) Write the member function Multiply to realize matrix multiplication. The parameter of this function is a reference to a Matrix object, and return a new Matrix object, which is the result of multiplying two matrices. If the matrix dimensions do not match, an empty matrix is returned.
- 10) Write the member function Transpose, which returns a new matrix, which is the transpose of the original matrix.
- 11) Requirements:
Fill in your code in the specified test Matrix.cpp, do not modify the main () function.
- 12) Submissions include :
 - A. For this experiment report, please replace ##### in the file name of the experiment report with your student number, and replace X XX with your name.
 - B. Source code. The above functions are written in the same source file, and the source file naming rule is C PP- 3 -#####.cpp, where ##### is replaced by your student number.

2. Paste the screenshot of the program running result below. (If the executable file cannot be generated due to grammatical errors, there will be no running results, and the screenshot will not be pasted)



```
Non rows = 0, cols=0

0.707107 -0
-0.707107 0.707107

-1.45682e+144
-1.45682e+144
```

3. Insert your completed source file below . (Insert method: first place the cursor at the beginning of the next paragraph of this paragraph, then select Insert->Object->Text in the file in the menu bar, and select the source file you wrote in the pop-up dialog box)

```
#include <iostream>

using namespace std;

/*Please change the following strings to your name and student number*/
char name[] = " XXX " ;
char ID[] = " AAAAA " ;

/* add your code below
The declaration and implementation code of the class are written below*/
class Matrix {
private :
    int rows; //Number of rows in the matrix
    int cols; //number of columns in the matrix
    int length;
    double *pValue; //The value of the matrix

public :
    /* Override the default constructor */
    Matrix() {
        length = 0;
        rows = 0;
        cols = 0;
        pValue = 0;
    }
    /*Constructor*/
    Matrix(const int numRows, const int numCols) {
        pValue = 0;
        rows = numRows;
        cols = numCols;
        length = rows * cols;
        if (length > 0) {
            pValue = new double[length + 1];
            pValue[length] = 0;
        }
        for ( int i = 0; i < length; i++)
            pValue[i] = 0;

    }

    /* Write the copy constructor below */
    Matrix & operator= ( const Matrix & s ) {
        if (& s != this ) {
```

```

        if (0 != pValue) {
            delete[] pValue;
        }
        pValue = 0;
        length = s.length ;
        if (length > 0) {
            pValue = new double[length + 1];
            memcpy(pValue, s.pValue, sizeof(double)*length);
            pValue[length] = 0;
        }
    }
    return *this;
}

/* Matrix(const Matrix &rhs) {
    rows = rhs.rows;
    cols = rhs.cols;
    if (rows > 0 && cols > 0) {
        pVale = new double(rows*cols);
        for(int i=0;i<rows*cols;++i)
    }
}

*/

/* Construction function */
~Matrix() {
    if (0 != pValue) {
        delete[] pValue;
        rows = 0;
        cols = 0;
        length = 0;
    }
}

/* Write other member functions below */
// assignment function
void SetValue( const int numRows , const int numCols , const double element ) {
    if (0 == rows || 0 == cols || nullptr == pValue) {
        cout << "Empty Matrix" << endl;
    }
    else
    {
        int i = 0;
        i = numRows *cols + numCols;
        pValue[i] = element;
        cout << endl;
    }
}
}

```

```

//value function
double GetValue(const int numRows, const int numCols)
{
    if (0 == rows || 0 == cols || nullptr == pValue) {
        cout << "Empty Matrix" << endl;
    }
    else
    {
        int i = 0;
        i = numRows * cols + numCols;
        return pValue[i];
        cout << endl;
    }
}

// fetch function
int GetRowCount()
{
    return rows;
}

//get column function
int GetColCount()
{
    return cols;
}

Matrix Multiply( Matrix & rhs ) {
    if (cols != rhs.rows)
        return Matrix ();
    else
    {
        Matrix result(rows, rhs.cols);
        for (int r = 0; r < rows; ++r) {
            for (int c = 0; c < rhs.cols; ++c) {
                double v = 0.0;
                for (int k = 0; k < cols; ++k) {
                    v += GetValue(r, k)*rhs.GetValue(k, c);
                }
                result.SetValue(r,c,v);
            }
        }
        return result;
    }
}

Matrix Transpose() {
    Matrix result(rows,cols);
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
            SetValue(i, j, GetValue(j, i));
}

```

```

        return result;

    }

    /*Display with no change*/
    void Display() const {
        if (0 == rows || 0 == cols || nullptr == pValue) {
            cout << "Empty Matrix" << endl;
        }
        else {
            int i = 0;
            for ( int r = 0; r < rows; r++) {
                for ( int c = 0; c < cols; c++) {
                    cout << pValue[i++] << ' ' ;
                }
                cout << endl;
            }
        }
    }
}

/* After the above code is written, please change the following macro to:
#define TEST 1
*/
#define TEST 1

/*===== */
/*Do not modify the following code*/
/*===== */
#define PI 3.1415926
;
int main( void ) {

    cout << name << " " << ID << endl;

#ifdef TEST
    Matrix Non;
    cout << "Non rows = " << Non.GetRowCount() << ", cols=" << Non.GetColCount() << endl;

    Matrix R(2, 2);
    double theta = PI / 4;
    R.SetValue(0, 0, cos(theta));
    R.SetValue(1, 0, -sin(theta));
    R.SetValue(0, 1, -R.GetValue(0, 1));
    R.SetValue(1, 1, R.GetValue(0, 0));
    R.Display();

```

```

    Matrix v(2, 1);
    v.SetValue(0, 0, sqrt(2.0) / 2);
    v.SetValue(1, 0, sqrt(2.0) / 2);

    Matrix w = R.Multiply(v);
    w.Display();

/*
    Matrix vt = v.Transpose();
    Matrix wn = R.Multiply(vt);
    wn.Display();
    */
#endif // TEST

    return 0;
}

```