Programming Practice: Matrix Class Inheritance

Design and implement a C^{++} project of matrices and vectors with class inheritance. The project has a basis class Matrix and two derived subclasses Vector and SMatrix, representing vectors and square matrices, respectively. Assume indices of rows and columns starting from 0. Class Matrix contians a matrix transposition operation. Specification of **class** Matrix in the header file matrix_inheritance.h is given as below:

```
class Matrix {
    friend ostream &operator<<(ostream&, const Matrix&); // friend output function
    friend istream &operator>>(istream&, Matrix&); // friend input function
    // element-wise scalar-matrix binary operations
    friend Matrix operator+(const double &, const Matrix &);
    friend Matrix operator-(const double &, const Matrix &);
    friend Matrix operator*(const double &, const Matrix &);

    protected:
      int row, col;     // row size and column size of a matrix
      double** m;    // a pointer to matrix elements
      void allocateMatrix();   // allocate matrix elements
      void deallocateMatrix();   // deallocate matrix elements
    public:
      Matrix(int=0, int=0);   // matrix constructor
      Matrix(const Matrix&);   // matrix copy constructor
      ~Matrix();   // matrix destructor

      void setSize(int, int);   // set the number of rows and columns in a matrix
      void setElement(int, int, double);   // set a matrix element
      double getElement(int, int) const;   // get a matrix element
      int getRow() const;   // get the number of rows in a matrix
      int getCol() const;   // get the number of columns in a matrix

      Matrix operator+(const Matrix &) const;   // matrix-matrix addition
      Matrix operator-(const Matrix &) const;   // matrix-matrix subtraction
      Matrix operator*(const Matrix &) const;   // matrix-matrix multiplication
      // element-wise matrix-scalar binary operations
      Matrix operator+(const double &) const;
      Matrix operator-(const double &) const;
      Matrix operator*(const double &) const;
      bool operator==(const Matrix &) const; // equal to relation for matrices
      bool operator!=(const Matrix &) const; // not equal to relation for matrices
      Matrix &operator=(const Matrix &);   // matrix assignment
      Matrix &operator+=(const Matrix &);   // matrix assignment with addition
      Matrix &operator-=(const Matrix &);   // matrix assignment with subtraction
      Matrix &operator*=(const Matrix &);   // matrix assignment with multiplication
      Matrix operator-() const; // interpret uniary operation as matrix transposition
      double determinant() const; // overloaded determinant function when row==col
    };
```

Subclasses Vector and SMatrix inherit class Matrix as shown below:

Vector: a single-column matrix, i.e., an n×1 matrix. Subclass Vector has a public member function for computing the inner product of two vectors with the same size. Let X and Y are two vectors of size n. The inner product of X and Y, denoted as X•Y, is defined $x_0y_0+x_1y_1+\ldots+x_{n-2}y_{n-2}+x_{n-1}y_{n-1}$.

$$X \cdot Y = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{bmatrix} = \sum_{i=0}^{n-1} x_i \times y_i.$$

Specification of **class** Vector in the header file vector_inheritance.h is given as the followings:

```
class Vector: public Matrix { // Inherit class Matrix
 public:
   Vector(int=0); // default vector constructor
   Vector(const Matrix&); // copy constructor from a column matrix
   Vector(const Vector&); // copy constructor from a vector
   void setSize(const int); // set the size of a vector
   double operator*(const Vector &) const; // use '*' for inner product operation
   Matrix operator-() const; // interpret unary operation as matrix transposition
};
```

SMatrix: a square matrix, i.e., an n×n matrix. Subclass SMatrix has a public member function for computing the determinant of the matrix. Let A be an n×n matrix square matrix The determinant is defined recursively as:

```
class SMatrix: public Matrix {
 public:
    SMatrix(int=0); // default square matrix constructor
    SMatrix(const Matrix&); // copy constructor from a matrix
    SMatrix(const SMatrix&); // copy constructor from a square matrix
   void setSize(const int); // set the size of a square matrix
   double determinant() const; // determinant function
};
```

Write the main program in matrix_inhericance_main_dxxxxxxx.cpp to perform the following operations:

1. Declare four Matrix objects A(4, 5), B(4, 5), C(4, 4), and D(5, 5), three Vector objects U(4), V(4), W(5), three Smatrix objects R(4), S(4) and T(5) and two scalar objects f and g of type **double**. Write function "void setMatrix(Matrix &);" to initialize elements of all Matrix, Vector, and Smatrix objects. For each element, its initial value is a random floating point number between -1 and 1 (including) with 4 digits after the decimal point. Also, input values of scalar variables f and g from the console. Note that, use function setMatrix(), instead of input stream cin>>, to initialize elements of matrix, vector, and square matrix objects.
2. Evaluate and print the results of expressions A+B, A+f, g+A, A+C, A-B, A-f, g-A, A-C, A*D, A*f, g*A, A*C, -A*B, and A*(-B). Print an error message and return the 0×0 matrix, if a matrix operation is invalid.
3. Evaluate and print the results of expressions -U*V, U*(-V), and U*V. Note that, evaluation of expression U*(-V) needs to cast U as Matrix type. Note that -U and -V denote transposition matrix of U and V, respectively.

4. Evaluate and print the results of expressions S*A, -A*S, S*(-S), (-S)*S, and A*T.

5. Verify and print the result |R|*|S|==|R*S| with error less than 0.0001.

6. Declare a Matrix object variable H. Execute statements "H = f*A+B*D;", "H=D-(-A)*C*B;", and "H=U*V*C*|T|;" and print the result of matrix H for each statement.