CIS-277 Data Structures and Algorithms

Programming Assignment: Abstract Data Type – Matrix

The purpose of this course is to provide you with experience in:

1. Identification of the appropriate ABSTRACT DATA TYPE (model and operations) to use for a given application.
2. Select of the best DATA STRUCTURE and corresponding ALGORITHMS to implement the ADT in a programming language.

There are many applications in mathematics, economics, engineering, etc. in which the underlying model is a matrix. Consequently, it would be useful to implement this ADT. The special case of a 2x2 matrix will be considered.

MODEL: the general form of a 2x2 matrix is:

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \qquad \text{example:} \qquad \begin{bmatrix} 2.5 & -3.7 \\ 4 & 0 \end{bmatrix}$$

Where: a11, a12, a21, and a22 are real numbers (scalars) that are the COMPONENTS (or ELEMENTS) of the matrix.

OPERATIONS: the following are some of the common operations that are performed on such matrices:

Matrix addition:

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} + \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix} = \begin{bmatrix} a11 + b12 & a12 + b12 \\ a21 + b21 & a22 + b22 \end{bmatrix}$$

Matrix subtraction:

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} - \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix} = \begin{bmatrix} a11 - b12 & a12 - b12 \\ a21 - b21 & a22 - b22 \end{bmatrix}$$

Scalar multiplication:

$$k \quad x \quad \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} = \begin{bmatrix} k \text{ x } a12 & k \text{ x } a12 \\ k \text{ x } a21 & k \text{ x } a22 \end{bmatrix}$$

Matrix multiplication:

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \times \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix}$$

$$= \begin{bmatrix} a11 \times b11 + a12 \times b21 & a11 \times b12 + a12 \times b22 \\ a21 \times b11 + a22 + b21 & a21 \times b12 + a22 + b22 \end{bmatrix}$$

Inverse of a matrix:

$$A = \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} a22 / \det(A) & -a12 / \det(A) \\ -a21 / \det(A) & a11 / \det(A) \end{bmatrix}$$

Where: det(A) is a scalar (the "determinant of A") computed as:

$$\det(A) = a11 \times a22 - a21 \times a12$$

(mathematical note: a matrix has an inverse only if det(A) is not zero. Do not test for this in your program).

In order to implement the MARRIX ADT in a programming language, it is necessary to first select an appropriate DATA STRUCTURE to represent the class. The data structure type name should be: MATRIX (that is, it should be possible to declare variables to be of type MATRIX).

After the data structure has been selected to represent the ADT, it then becomes necessary to design the functions that will implement the corresponding OPERATIONS. <u>Write the complete definition code for the following functions:</u>

get_matrix(name, m);	enables the user to enter ONE matrix name and ONE matrix.
get_scalar(b);	enables the user to enter ONE scalar.
calc_sum(m1, m2, sum);	computes: m1 + m2
calc_diff(m1, m2, diff);	computes: m1 – m2
scalar_mult(k, m, k_m);	computes: k x m
calc_prod(m1, m2, prod);	computes: m1 x m2
calc_inv(m, m_inv);	computes: the inverse of m (the determinant is a local variable)
print_matrix(outfile, name, m);	prints ONE matrix in an appropriate form and a name that identifies it.

Be certain that you make proper use of the data structure: MATRIX. Note that these functions would form the basis of a "toolkit" for the MATRIX ADT. This toolkit could then be used to quickly write APPLICATION PROGRAMS that involve the use of 2x2 matrices.

Prior to using the MATRIX toolkit in applications programs, each of the functions in it should be tested to be certain that they work properly. Therefore, you are to write a "throw away driver program" that uses each of the functions in this toolkit. The program should:

a. Request the user to input two matrices: mq and m2, and their names (for the test run, the names can be just: m1 and m2). This will require TWO SEPARATE calls to the get_matrix function.
b. Request the user to enter a scalar value.
c. Compute: m1 + m2, m1 – m2, k x m1, m1 x m2, and m1 inverse.
d. Print out the original two matrices, each of the resulting matrices, and their corresponding names (sum, difference, etc.). This will require several separate calls to the print_matrix function.

<u>DIRECTIONS:</u> USE THE FOLLOWING MATRICES AND SCALAR FOR THE SAMPLE RUN:

$$m1 = \begin{bmatrix} 7 & 2.5 \\ -3.0 & 2.0 \end{bmatrix} \qquad m2 = \begin{bmatrix} 8 & 6.5 \\ 5.0 & 1.3 \end{bmatrix} \qquad k = 4.5$$

<u>Sample Output:</u> the output form should be similar to either one of the following:

```
                15.0    9.0
     sum =
                2.0     3.3

             _                 _
            |   15.0    9.0     |
     sum =  |                   |
            |_  2.0     3.3    _|
```

<u>Programming Notes</u>

1. Use a STEPWISE MODULAR APPROACH in the development of this program (and all future programs):

   Design and implement ONE FEATURE of the program; then add another feature; continue adding ONE FEATURE at a time until the program is complete.

   Example: begin by implementing the following:

   ```
   get_matrix(name1, m1)
   get_matrix(name2, m2)
   ```

```
calc_sum(m1, m2, sum)

open_file(outfile)
print_matrix(outfile, name1, m1)
print_matrix(outfile, name2, m2)
print_matrix(outfile, “Sum =“, sum)
close_file(outfile)
```

<u>Submit the following</u> (in the order specified):
1. A source code listing
2. The program output.
3. The handwritten work.