EECS 10: Assignment 6

November 1, 2004

Due Wednesday 11/17/04 at 12:00pm

1 Matrix Calculation [80 points + 10 bonus points]

A matrix is a two-dimensional data structure. It is usually organized as a row of columns of numbers. Examples of matrices are:

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 2 & 4 & 5 \\ 1 & 1 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 6 & 2 \\ 2 & 5 & 3 \\ 1 & 6 & 7 \end{bmatrix}$$

A and *B* are examples of 3 by 3 matrices. Each element of the matrix can be addressed by the row, column indices, mathematically written as a(i,j) where i and j are the row and column indices, respectively. I your program, you will use two-dimensional arrays of floating point values to represent such matrices, i.e. the matrix A should be defined as `double A[3][3];`

Your program should be a menu driven program where always the current matrix A is displayed and the user then selects matrix operations from the following menu:

```
     |  1.0   0.0   0.0 |
 A = |  0.0   1.0   0.0 |
     |  0.0   0.0   1.0 |

(1)  Set matrix A
(2)  Add a matrix B (matrix addition)
(3)  Subtract a matrix B (matrix subtraction)
(4)  Multiply with a matrix B (matrix multiplication)
(5)  Scale matrix A (scalar multiplication)
(6)  Compute the determinant of matrix A
(7)  Transpose matrix A
(8)  Invert matrix A (compute the inverse matrix)
(9)  Exit

Please enter your choice:
```

Your program should perform the following operations for the options:

(1) Enter matrix A
This function should allow the user to set the value of A by entering a number for each element of the matrix.

(2) Matrix Addition
Here, a second matrix B should be entered by the user. Then, the program should compute the addition of the matrices A + B and assign the result back to matrix A so that the result of the matrix addition is displayed before the next menu. (If you prefer, you may store the result of the addition first in a local matrix C, and then copy C over to A).

Hint: Matrix addition is defined as follows:
C(i,j) = A(i,j) + B(i,j) for all values of i and j

(3) Matrix Subtraction
This should the very same procedure as for option (2), except that the matrix B is subtracted from matrix A.

(4) Matrix Multiplication
Again, this is very similar to option (2). Here, however, the matrix A should be multiplied by matrix B by use of a matrix multiplication.

Hint: Each element in the result of a matrix multiplication A * B is computed as the sum of the products of the row elements of A and the column elements of B.

(5) Matrix Scalar Multiplication
Let the user enter a scalar value *s* by which the matrix A should be scaled.

Hint: For a scalar multiplication, simply multiply each matrix element with the scalar number.

(6) Determinant of the Matrix
Compute the determinant of the matrix A and display it (the determinant is a scalar number).

Hint: The determinant for a 3 by 3 matrix is defined as follows:

```
Determinant =   A(1,1)[A(2,2)*A(3,3)-A(2,3)*A(3,2)] +
                A(1,2)[A(2,1)*A(3,3)-A(2,3)*A(3,1)] +
                A(1,3)[A(2,1)*A(3,2)-A(2,2)*A(3,1)]
```

(7) Transpose of the Matrix
Compute the transpose of matrix A.

Hint: The transpose of a matrix is obtained by swapping the row and column indices, i.e.
T(j, i) = A(i, j) for all i and j

(8) Matrix Inversion (Extra credit)
Compute the inverse of matrix A. Note that the inverse matrix does not always exist. Your program should handle this case in some well-behaved manner.

Hint: The result of multiplying a matrix with its inverse matrix is the identity matrix where every element is 0 except for the main diagonal elements which are 1.

(9) Exit
This option will terminate your program.

Implementation

To implement your program, you should use separate functions to perform the operations required by the menu options. Name your functions properly. The function names should detail what the function does (e.g. DisplayMatrixA( ) would display the matrix A, AddMatrixBtoA( ) would perform the matrix addition, etc.).

Define all matrices as global variables so that all functions have easy access to them.

There should be three matrices used in your program:

- Matrix A: used as the current matrix and the one on which all operations are performed. The result of each operation should also be stored in matrix A for use in the next operation. At the beginning of your program, this matrix should be initialized to the identity matrix (as shown at the beginning).

- Matrix B: used as the operand matrix. That is, when performing the addition, subtraction, etc., matrix A should be the first, and matrix B should be the second operand. This should be consistent throughout the program.

- Matrix C: used as a temporary matrix. You may temporarily store the result of a matrix operation in C. However, before returning to displaying the menu options, matrix C should be copied over into A so that the result can be used for further operations.

Script file

To demonstrate that your program works correctly, perform the following steps and submit them as your script file:

1. Set matrix A to the following: $A = \begin{bmatrix} 2 & 1 & 4 \\ 1 & -1 & -2 \\ 0 & -3 & 3 \end{bmatrix}$
2. Add the following matrix B to A: $B = \begin{bmatrix} 2 & -3 & -5 \\ 2 & 3 & 2 \\ 1 & 3 & -3 \end{bmatrix}$

3. Subtract the following matrix B from the result: $B = \begin{bmatrix} 1 & 3 & 1 \\ 3 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$
4. Multiply the resulting matrix with the following: $B = \begin{bmatrix} 0 & 0 & 1 \\ 3 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}$
5. Scale the resulting matrix A by 1/4.
6. Compute the determinant of scaled matrix A.
7. Transpose matrix A.
8. For extra credit:

   Compute the inverse of the following matrix: $A = \begin{bmatrix} 1 & -3 & 1 \\ -1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$

   Then, to verify that the result is actually the inverse matrix, multiply it with the original matrix A and check if the result is the identity matrix.
9. Exit the program.

What to submit

Use the standard submission procedure to submit the following files:

- matrix.c
- matrix.script
- matrix.txt