Matrices Program Assignment
Numerical Analysis

Goal: Create a project which can do simple calculations with matrices.
Steps to follow - in this order:

1. Open the programming project:
   - Unzip the MatrixApplication file linked from eclass. Simply double clicking the .zip file to open it will NOT suffice. It will not work properly in Visual Studio if you do this.
   - Double click the MatrixApplication.sln file. This should open the project up in Visual Studio. You will need to use version dotnet 8.0 for all your programs this semester.
2. Identify and inspect the interface.
   - You will find the Solution Explorer on the right hand side bar of Visual Studio. Use the Solution Explorer to find the interface which you need to implement. Recall that all interfaces are named with a leading capital I. So the interface in this project is a file named IMatrixObejct.cs. You should NOT make any changes to this file. However, you can inspect the interface to see what you are supposed to do in your implementation. You can open the interface (or any file in the file explorer) by double clicking the name of the file in the Solution Explorer.
3. Add the implementation file to the project
   - Right click the name of the project (the bold name in the Solution Explorer) and select ADD and then in the sub-menu, select Class...
   - Select the Interface option and then, in the bottom area of this window pane, name the class the same name as the interface, except without the leading I. So this class should be named MatrixObject.cs
   - Click Add
4. Setup the inheritance
   (a) If the implementation class you just created does not automatically open then double click the its name in the Solution Explorer to open it.
   (b) Replace the word internal with the word public. This changes the class to a public class which has greater accessibility within the project.
   (c) Setup the inheritance from IMatrixObject.cs by adding :IMatrixObject to the end of the line "public class MatrixObject" like you learned to do in one of the tutorials.
   (d) Click the little light bulb - usually on the left hand side of the screen - and tell it to implement the interface. This should copy all the functions that I specified in the interface to your implementation and give you space to write your code.
5. Determine what you need to program by reading the interface.
   - The function Add returns a doubly indexed array (which is used to store a matrix) of type double. It accepts two doubly indexed arrays A and B each of type double. Your program should use the correct rules for adding two matrices together and return the sum of these two matrices. Keep in mind that the matrices must be the same shape in order to add them. Therefore your program must check that they are the same shape before adding them. If they are not the same shape, your program should throw an exception stating that they are not the same shape.
   - The function Subtract returns a doubly indexed array (which is used to store a matrix) of type double. It accepts two doubly indexed arrays A and B each of type double. Your program should use the correct rules for subtracting two matrices and return the difference of these two matrices. Keep in mind that the matrices must be the same shape in order to subtract them. Therefore your program must check that they are the same shape before subtracting them. If they are not the same shape, your program should throw an exception stating that they are not the same shape.
   - Multiplication: There are two versions...

  - multiplication of two matrices - this is the first of the multiply functions. This function returns a doubly indexed array of type double. It accepts two doubly indexed arrays A and B of type doubles. Your program should use the correct rules for multiplying matrices to calculate the product of these two matrices and return this product as the answer. Make sure that you check that the matrices have the correct dimensions in order to multiply them. If they do not have the correct dimensions, your program should throw an exception stating that the matrices cannot be multiplied together.
  - multiplication of a matrix and a constant - this is the second of the multiply functions. This function returns a doubly indexed array of type double. It accepts a doubly indexed array A and a constant c of type double. Your program should use the correct rules for multiplying a matrix and a constant to calculate the product and return this product as the answer.
- The function Gauss does not return anything. It accepts a doubly indexed array A of type double and a single array l of type int. Gauss completes the elimination on A and updates l and A.
- The function GaussModifiedForwardEliminationRHS does not return anything. It accepts a doubly indexed array A of type double, a single array l of type int, and a single array b of type double. This part of the program implements the forward elimination information and applies it to the b column (the b column talked about in the textbook).
- The function Solve returns an array of type double. It accepts a doubly indexed array A of type double, a single array l of type int, and a single array b of type double. This function applies all the information of the elimination process to the x column vector and returns this as the solution. This is the last part of the Solve Procedure Pseudocode in the textbook.
- The CalculateMatrixInverse function returns a doubly indexed array of type double and accepts a doubly indexed array A of type double. It uses the Gaussian Elimination functions to calculate the inverse of the matrix A and returns this inverse matrix.

6. You can use the Program.cs file to put all your I/O. This can be used to help debug your program.

7. Submit your program

   - Create a zip file of your MatrixObject.cs file.
   - Using the link to OICLearning posted on eclass, navigate to the Matrix Application assignment.
   - Submit this zip file on OICLearning. Wait a little while (up to a few minutes) to see your grade. You can resubmit as much as you want until the deadline is passed. I will record your last score on eclass - so make sure that this is your highest score.

8. **Coding Hints for Inverses:** Note that you are just doing Gaussian Elimination with two matrices at the same time. So, do the following:

   (a) Input your matrix A.
   (b) Check to see that matrix A is square.
   (c) Create a square identity matrix with the same size as A.
   (d) Call A.GaussianElimination();
   (e) for each column (starting with the first column)
      i. set BColumn = this column of the identity matrix
      ii. call A.GaussianEliminationRHS()
      iii. call A.backsubstitution()
      iv. copy your answer from A.X into this column of your inverse matrix