CE/CS 1337 – PROJECT 2 – Don't Cross the Streams

Pseudocode Due: 2/19 at 11:59 PM

Project Due: 3/9 at 11:59 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- All project deliverables are to be submitted in eLearning.
- The pseudocode should be submitted as a Word or PDF document and is not accepted late.
- Zip all of the source files into a single zipped file
    - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
    - Please review the submission testing information in eLearning on the Course Homepage
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile using gcc 7.3.0 or higher with the following flags enabled
    - -Wall
    - -Wextra
    - -Wuninitialized
    - -pedantic-errors
    - -Wconversion
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted. (-5 points)**

Objective:

- Allocate memory dynamically.
- Utilize pointers to directly manipulate memory.

Problem: You have been hired by EA to work on a new Ghostbusters game. In this game, there is a multiplayer mode where players must capture the ghosts and as all good Ghostbusters know you can't cross the streams. Your task is to create the framework to identify at what point the streams might cross in a 3-dimensional space. You will do this by implementing the Gauss-Jordan Elimination method of solving a linear system of equations.

Pseudocode: Your pseudocode should describe the following items

- Main.cpp
    - Detail the step-by-step logic of the main function
    - List other functions you plan to create
        - Determine the parameters
        - Determine the return type
        - Detail the step-by-step logic that the function will perform

Details

- You will utilize a dynamically allocated array
    - You may create a dynamic multi- or single dimension array
    - Dynamic multi-dimensional arrays require double pointers which we will not cover in class
    - Remember that standard multi-dimensional arrays in C++ are stored as single dimension arrays in memory
        - This may be easier to work with than the double pointers.
- The number of rows will be dependent on the number of players (2-4 players)
- The number of columns will always be 4 and represent x, y, z, and a constant (always in that order)
- The size of the array should be determined by the number of equations in the file.
- You are not expected to write a program that solves the matrix. We will leave it to someone else at EA to develop the AI that utilizes your framework and solves the matrix automatically. We are simply providing the functions that allow changes to be made to the array.
- The linear equations will be read from a file
- Develop a user interface to modify the equations
- Row options available to the tester are:
    - Switch 2 rows
    - Multiply a row by a non-zero number (positive or negative)
    - Add a scalar multiple of one row to another row
- If the matrix is in reduced echelon form (http://mathworld.wolfram.com/EchelonForm.html), do not allow the tester to continue row operations.
- If a row has been solved (all numbers on the row are 0 or only one of the variable columns contains a 1), the row cannot be multiplied by a non-zero scalar (row option 2 above).
- All interaction with the array must be done using pointers and pointer arithmetic. That means no bracket notation in this project except to create the array and no offset notation. (-20 points if bracket notation used, -10 points if offset notation used)
- Floating point numbers will be displayed to 2 decimal places

Input and User Interface:

- Develop an easy to use menu system for the tester.
    - 1 – Switch two rows
        - Prompt the user for the number of the rows to switch
    - 2 – Multiply row by non-zero number
        - Prompt for row and number to multiply
    - 3 – Add scalar multiple of one row to another row
        - Prompt for the following in the exact order listed
            - Row to use for multiplication
            - Multiplier
            - Row to be modified by adding scalar multiple
    - 4 – Quit
        - Exits the program
- All input will be entered from the keyboard and will be of the valid data type.

    - You do not have to worry about the tester entering a letter when a number is expected or a floating point number when an integer is expected.
- You are expected to verify input is within range
    - If a number out of range is entered, loop until a valid number is entered

Equation Input:

- Equations will be read from a file named `matrix.txt`.
- The file will contain equations for each player
    - The file will have two, three or four equations
- Each line in the file will end with a newline (except the last line which may or may not have a newline)
- Each equation will consist of up to 3 variables and a constant.
- The variables will always be x, y and z
- The variables may have a coefficient
    - The coefficients can be floating point numbers
- If a variable does not have an explicit coefficient, it is assumed to be 1
- If a variable is missing from the equation, the coefficient is assumed to be zero
- There will be no spaces in the equations
- All variables will be to the left of the equal sign
- The variables can be in any order
    - Examples
        - `3.5x+2.95y+.3z=-2`
        - `z+5y-3x=10`

Output:

- After every row operation, display the modified matrix to the console.
- If the matrix is in reduced echelon form, display the variables and their known values as illustrated by the matrix.
    - Remember that in a 2x4 matrix only 2 of the variables can be solved through Gauss-Jordan Elimination. You are not required to calculate the third variable in this situation.
    - In a 4x4 matrix, all variables must be solved, and the last row must be all zeroes to be in reduced echelon form