Homework 3: Functions, loops, and decisions **Due Date:** 11/13/24

A **rational number** is a number that can be expressed as a fraction of the form **a/b**, where **a** is the **numerator** and **b** is the **denominator** and both are integers. Arithmetic operations on two rational numbers a/b and c/d are defined as follows:

Addition: $\frac{a}{b} + \frac{c}{d} = \boxed{\frac{(a * d) + (c * b)}{(b * d)}}$

Subtraction: $\frac{a}{b} - \frac{c}{d} = \boxed{\frac{(a * d) - (c * b)}{(b * d)}}$

Write an interactive program that presents the following menu to the user:

Rational numbers calculator

(A)ddition
(S)ubtraction
(Q)uit

Enter your option:_

The menu options must be displayed by a void function named **showMenu()**. This function **just** shows the **title** and the **options**, **the decision whether to call add(), subtract(), or quit must be done in main()**.
The Addition and Subtraction **options** must be implemented by a couple of void functions named **add()** and **subtract()** respectively.
These functions must clear the screen and show a title specifying the operation to be performed. Next, they have to show a couple of input prompts asking for the rational numbers and, after getting and processing them, they have to display the result.
After performing their corresponding operations, the functions must ask the user whether they would like to do more operations. If the user chooses to do more operations, the functions repeat the steps described in the previous paragraph (that is, they clear the screen, ask for both numbers, and so forth). This process is repeated for as long as the user chooses to do more operations. When the user is done with an operation, the function must be exited and the menu must be shown again.
All the letters entered by the user as an answer to a question posted by the program must be accepted in upper or lower case.

The screen shown by these functions should look like this:

Addition of rational numbers

Please enter a fraction (n/d): **1/2** (entered by the user)

Please enter a fraction (n/d): **2/4** (entered by the user)

The result of 1/2 + 2/4 = 1

Do you want to do more additions? (Y/N):

The different operations on the rational numbers must be implemented using the following functions:

```
      GetRational(int num, int den);
// This function shows the prompt Please enter a fraction (n/d): , gets the values
// from the keyboard and passes them to the caller. Since the division operator (/)
// is read in along with the numbers the function must read and discard it.
// This function must reject a denominator equal to zero.

      AddRational(int anum, int aden, int num1, int den1, int num2, int den2);
// num1, den1, num2, and den2 are received from the caller.
// anum and aden are calculated using the formula shown above and passed to the
// caller after they are reduced. To reduce the fraction this function must call
// the function reduce(anum, aden).

      SubtractRational(int anum, int aden, int num1, int den1, int num2, int den2);
// num1, den1, num2, and den2 are received from the caller.
// anum and aden are calculated using the formula shown above and passed to the
// caller after they are reduced. To reduce the fraction this function must call
// the function reduce(anum, aden).

      DisplayRational(int num, int den);
// num and den are received from the caller.
// The numbers are shown in the form num/den unless the denominator is equal to 1
// in which case only the numerator should be displayed.

      reduce(int num, int den)
// num and den are received from the caller.
// num and den are reduced according to the algorithm explained below and passed back
// to the caller.
```

All the above functions (except reduce() which is called by AddRational() and SubtractRational()) must be called by **add()** and **subtract()** to perform the corresponding operations.

IMPORTANT:

- Do NOT remove or modify the statements that I use to test certain things in your program.
- You must define whether each function is a void or a value-returning function.
- You must define whether each parameter is a value parameter or a reference parameter.
- You must use the most appropriate type of loop and decision statements.
- You must use the identifiers I provide you for the parameters and functions.
- **You must use the following algorithm to reduce the fraction.**

To reduce a fraction, you need to find the **greatest common factor** (**gcf**) of the numerator and denominator. Once it is found, divide both the numerator and denominator by the **gcf**.

The greatest common factor of two integers is the largest integer that will divide both numbers evenly. For example, the greatest common factor of 18 and 27 is 9, since 9 is the largest integer to divide evenly into both 18 and 27. Since 9 goes into 18 twice and into 27 three times, the reduced fraction is 2/3. An efficient algorithm for computing the gcf is attributed to the Greek mathematician Euclid (ca. 300 B.C.).

Euclid's algorithm:

```
1.    let A and B be two positive integers (Hint: use abs())
2.    let  R = remainder after dividing  A by B
3.    while  R != 0
            A = B
            B = R
            R = remainder after dividing A by B
4.    B  is the gcf
```

Example: suppose you need to reduce the fraction 24/60

First you find the gcf of 24 and 60:

```
          A = 24, B = 60          24/60 is 0 with remainder R = 24
     set  A = 60, B = 24          60/24 is 2 with remainder R = 12
     set  A = 24, B = 12          24/12 is 2 with remainder R = 0
```

So, the **gcf** is **12**. Then, you need to divide 24 by 12 and 60 by 12 to get the fraction reduced. The resulting fraction is 2/5.

For your reference, I am providing a video in which I show how the program should work. Test your program with the different sets of input values shown below and compare your results with mine to ensure that your program is working correctly (pay attention to the output generated by both programs).

https://utrgv.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=abd1f4b6-20c2-47f5-b9f4-add30188159b

Test plans:

Set of inputs to be used to test the program:

A) 1/2 and 2/4
B) 1/3 and 3/4
C) 9/5 and 3/0 (entry 3/0 should be rejected so reenter 2/5)
D) 9/0 and 3/4 (entry 9/0 should be rejected so reenter 6/8)

Use them to test **both** operations (**add** and **subtract**).

To clear the screen, use the function shown below:

```
Void clear_screen()     // This function clears the screen
{
      cout << "\033[0;0H\033[2J";
}
```

To pause your program, use the function shown below:

```
void pause()      // This function pauses the execution of the program
{
      cout << "Press Enter to continue ...";
      cin.sync();
      cin.ignore();
      cin.get();
}
```

The program must compile without errors or warnings.
Open **hw3.cpp** in your IDE and implement your solution.
Review the examples discussed in class, the lab assignments done so far, and the textbook to get an idea of what you need to do. The **algorithm** must be written in

pseudocode and should look like my lab handouts. **Include your algorithm in the source code as comments.**

Your program must be well commented, use meaningful identifiers, **use efficient control structures**, and use indentation as shown in the textbook.
Your program must have the following header:

```
//*************************************************************************
// Team:           CSCI 1470                Fall 2024             Homework # 3
// First and Last Name
// First and Last Name
//  Using your own words, write here a description of what the program does.
//
//*************************************************************************
```

When done, submit your solution through Blackboard using the "Assignments" tool. Do Not email it.

Paste the link to your final solution along with your source code in the textbox opened when you click on Create Submission before you click on Submit.

Grading criteria

You start with 100 points and then lose points as you don't do something that is required.

-10: fractions are not reduced **before** they are returned by addRational() and subtractRational()
-20: missing/poor algorithm
-05: source code is not well indented (formatted)
-10: missing/too few comments
-10: didn't follow the specified format for input/output.
-15: didn't implement the required functions (each)
-10: wrong implementation of a required function (each)
-05: didn't accept answers in upper or lower case (use of toupper(), tolower(), or the OR operator (||) are accepted)
-10: didn't use value and reference parameters or function value data type correctly (each function)
-05: incorrect function name (each function)
-10: incorrect function call (each function)
-10: wrong implementation of main()
-20: didn't use the most appropriate loops where needed
-20: didn't use the most appropriate decision statements where needed
-10: didn't reject a denominator equal to zero
-50: program doesn't compile
-40: program doesn't run as expected
-20: didn't pass all my tests
-50: Incomplete program.
-20: Incorrect/missing source code or incorrect/missing link to your solution
-100: No team contribution.
-100: The code submitted is not your creation (you got it from a web site or another person)
-100: No submission.
-10: Late submission.

Important: more points may be lost for other reasons not specified here.