CSCI 110 -- Project 2 (Fraction Tutor)

Due Tuesday, 11/21/23

Write a tutorial program that will provide the user with a fraction arithmetic package. Your program must be able to perform addition, subtraction, multiplication, and division with fractions.

Here are the requirements for your program:

1. The computer must select a random problem from the list of available problems that was provided in `P2Fractions.txt` and you can assume that there will be no more than 50 problems in the input file.
2. There are two available modes: training mode (correct answer is provided ahead of time) and normal mode (correct answer is not provided ahead of time).
3. The program must be interactive; the flow of the program should go as follows:
    - There are 5 problems per session.
    - At the start of each problem, display the problem and answer if applicable (training mode).
    - At the beginning of each problem, you should display the number of attempts left.
    - Ask the user to supply an answer in reduced form.
    - The user should receive feedback immediately after each attempt.
4. Some additional rules for the current version of this program:
    - A user is allowed up to 3 attempts. Make sure to remind the user of how many attempts s/he has left.
    - Assume that user will only submit valid response (numerator/denominator).
    - The problem should end when the user enters the right answer or runs out of attempts. If the user runs out of attempts, reveal the correct answer.
5. Only one session per run unless extra credit is implemented.

The output of a correct attempt should look like this, but number of problems and actual problem could be different:

```
Welcome to the Fraction Tutor V1 by Your Name!

There are 5 problems per session.  The computer will select a problem
from a list of problems and you have up to three attempts per problem.
You must provide an answer in reduced form (i.e., enter 3/4 instead of 6/8).
Have fun and good luck.

Loading problems from file P2Fractions.txt ...
There are 26 problems available.

Available Modes
   1. Training mode
   2. Normal mode

Please select a mode: 1<Enter>
Training mode is selected.
```

```
Computer is selecting a random problem ...

Problem 1: 1/2 + 1/4
Correct answer: 3/4
---------------------------------------------
You have 3 attempts left.
Please enter your result: 6/8<Enter>
It is incorrect.
---------------------------------------------
You have 2 attempts left.
Please enter your result: 1/2<Enter>
It is incorrect.
---------------------------------------------
You have 1 attempt left.
Please enter your result: 3/4<Enter>
Congratulations!  It is correct.

Computer is selecting a random problem ...

Problem 2: 1/2 - 1/4
...

You answered 4 out of 5 problems correctly (80.0%).
Your grade is B.
```

And the output of an incorrect attempt should look like this (instructions omitted):

```
Available Modes
   1. Training mode
   2. Normal mode

Please select a mode: 2<Enter>
Normal mode is selected.

Computer is selecting a random problem ...

Problem 1: 1/2 - 1/4
---------------------------------------------
You have 3 attempts left.
Please enter your result: 6/8<Enter>
It is incorrect.
---------------------------------------------
You have 2 attempts left.
Please enter your result: 1/2<Enter>
It is incorrect.
---------------------------------------------
You have 1 attempt left.
Please enter your result: 3/4<Enter>
It is incorrect.
---------------------------------------------
Sorry, you are out of attempts!
Correct answer: 1/4

Computer is selecting a random problem ...

Problem 2: 1/2 * 1/4
...

You answered 3 out of 5 problems correctly (60.0%).
Your grade is D.
```

You need to come up with a design for *main()* and other modules/functions. See the two modules/functions below that you must use in your program. **You must properly utilize modules/functions to receive full credit for this project.** The `main()` function should display information about the program, loads the problems from a file into an array of strings, allows user to select a mode (training or normal), and then runs the loops for the number of problems per session. For each problem, the computer generates a random number to pick a random problem from the list. This starts up an interaction between the user and the computer. Be sure to allow multiple attempts if needed and keep track of correct responses. Display the results once the session is over.

Implement the module (C++ void function) `loadProblems` that has two parameters – an array to hold a list of problems and a pass-by-reference integer to hold the count. This module sends back a count (by reference) representing the number of problems and it also puts problems in the original array. Next, implement the function `gcd` that takes in two parameters. This function returns the great common denominator of two positive integers. You can use the result of this function to reduce a fraction.

Here are the two function prototypes:

```
        void loadProblems(string problems[], int &count);
        int  gcd(int a, int b);
```

You can use the following module (void C++ function) to convert a problem in string to the appropriate fractions and operator:

```
void convertProblem(string problem, int &num1, int &den1, char &op,
                         int &num2, int &den2)
{
    char slash;                      // input char / between numerator and denominator
    istringstream iss(problem);      // need to include <sstream>

    iss >> num1 >> slash >> den1;    // first fraction
    iss >> op;                       // operator
    iss >> num2 >> slash >> den2;    // second fraction
}
```

Extra credit: You can earn up to 4 additional points if your program allows multiple sessions (use Y/N loop) and keeps track of overall score. Print the overall score before exiting the program. In addition, **the computer must select a new problem every time for each session** (i.e., if a problem was used already for that session, don't use it again for that session so you need to implement some logic to ensure that is the case; however, it can be used again for a new session). **You can include extra credit features in the same version (no need for a separate version)**. See sample output below (some information is omitted):

```
        Available Modes
           1. Training mode
           2. Normal mode

        Please select a mode: 2<Enter>
        Normal mode is selected.

        Computer is selecting a random problem ...

        Problem 1: 1/2 + 1/4
```

```
---------------------------------------------
You have 3 attempts left.
Please enter your result: 3/4<Enter>
Congratulations!  It is correct.

Computer is selecting a random problem ...

Problem 2: 1/2 - 1/4
...

You answered 5 out of 5 problems correctly (100.0%).
Your grade is A.

Do you want to try another session: Y<Enter>

Computer is selecting a random problem ...

Problem 1: 1/2 / 1/4
. . .

Do you want to try another session: N<Enter>
Quitting the program.
You answered 9 out of 10 problems correctly (90.0%).
Your grade is A.
```

Please provide documentation and apply good coding style because it is part of the grade. You must come up with enough test cases since the test cases are also part of the grade. Please submit the following items as a pdf file on Canvas using the project submission file:

1. Title page with name, class, project number, and relevant information about your program (compiler and system used, file names).
2. Status and notes about your program (status of your program at the minimum).
3. Hierarchy chart and pseudocode.
4. Input/output (copy and paste).
5. Source code (copy and paste).

In addition, you must submit a copy of your source code on Canvas (.cpp file only). Therefore, you must submit at least two files: pdf file and cpp file. You would need to submit one more cpp file for extra credit.

Your program will be graded as follow:

- Correctness: 25 points
- Test Cases (input and output): 5 points
- Design Documentation (hierarchy chart and pseudocode): 5 points

Documentation (pdf file and source code) and Coding Style: 5 points