CS 135 - Project 2 -- Math Skills Practice Program

INTRODUCTION

The purpose of Project 2 is to practice the skills that you are learning in Chapters 1-6 of the textbook, with emphasis on Loops and Files (Chapter 5); and Functions (Chapter 6).
The program shall test a user's math skills with randomly generated arithmetic problems.
The program shall be written for a 5th grade audience.

INSTRUCTIONS

Write a program that creates random arithmetic problems for solution. You may use any of the procedural programming code we developed in class, but you are instructed to write your own program.
When you are finished and want to test your program, document your tests by taking screenshots:
    Use *Alt+PrtScn* to copy the active window, and paste it into an MS-Word document.
When making your Word doc, include a sentence for each screenshot that tells what is going on in it:
    what the test was; what the result was; etc.
Your Word doc should include at least 15 screenshots, along with associated description.

Main Menu

There shall be one main menu, as follows:

```
                 ARITHMETIC PRACTICE PROGRAM

        Welcome to the Math Skills Practice Program.
        This program allows you to practice your math skills.
        Choose what to practice in the menu shown below.

        ---------------------------------------------------------
                               MAIN  MENU
        ---------------------------------------------------------

                       a.   Addition
                       b.   Subtraction
                       c.   Multiplication
                       d.   Division
                       q.   Quit the program

        ---------------------------------------------------------

        Enter your choice [ a b c d q ]:
```

Difficulty Level and the Role of the Random Number Generator

The "difficulty level" chosen by the user shall determine the number of digits used for each operand in a problem. The program shall offer difficulty levels 1 through 5 to the user at program startup. The random number generator (RNG) shall be used to generate the operands for each problem. For example, if a user chooses level 3 and the current problem is a multiplication one, then a problem like the following should be generated.

```
        Solve:

               123
             X 456
             -----

        (a)  56088
        (b)  57045
        (c)  21321
        (d)  14198
        (e)  NONE OF THE ABOVE
```

The numbers "123" and "456" are the operands for the problem, and the RNG is responsible for selecting them. In the case of divison, the two random numbers shall be the divisor and the quotient.

The RNG shall also generate the five choices for the problem, and decide which of them is the correct one. If the correct choice is not NONE OF THE ABOVE, exactly one of the remaining four choices must be the correct answer. If the correct choice is NONE OF THE ABOVE, the RNG must select four wrong answers for 'a' through 'd'.

Output file

The name of the output file shall be: ***firstname_lastname_results.txt***, using the user's *actual* first and last name. The output file shall consist of:

ITEM 0	User's name
ITEM 1	Timestamp for program start
ITEM 2	The number used to seed the random number generator
ITEM 3	The difficulty level chosen by the user

The following items relate to a single problem and will be repeated, as a group, for each problem given to the user:

ITEM 4	Timestamp for the problem
ITEM 5	Problem shown to the user (includes all choices presented to the user)
ITEM 6	Correct choice for the problem
ITEM 7	Number of attempts by user
ITEM 8	The choice given by the user
ITEM 9	User's current "correct" percentage

At the termination of the program:

ITEM 10	Timestamp for program end

Program Design

Here is a rough program design. You will need to flesh it out.

```
STEP 1.   Startup details
          ---- Get user name
          ---- Get difficulty level
          ---- Seed RNG
          ---- Open output file
STEP 2.   Display main menu
STEP 3.   Get user's main menu choice
          ---- Quit if necessary
STEP 4.   Generate and present problem to user
          ---- Interact with user
STEP 5.   Allow three chances to get the correct answer
          ---- Interact with user
STEP 6.   Record results to output file
STEP 7.   Return to STEP 2
```

It should be apparent that STEPs 2 through 7 form a loop.

Upon quitting, the program shall write its final information to the output file, then close it.

PROGRAM REQUIREMENTS

The program shall be menu-driven, and the user shall be prompted for information that is needed.
The program shall present arithmetic problems for solution that are appropriate for a 5th grade audience.
The user shall be given three attempts to solve each problem.
Each problem shall have five (5) choices: 'a' through 'e'.
In each case, choice 'e' shall be NONE OF THE ABOVE.
The "difficulty level" chosen by the user represents the number of digits used for the problems.
The random number generator (RNG) shall be used for all problems.
An output file shall be produced with the data described above.
The program shall be written using procedural programming methods and modular design. (No "goto's" or OOP.)
All detailed code shall reside in functions other than *main()*.
No global variables are to be used:

Data values must be passed into and out of functions using the techniques of Chapter 6 in our textbook.

An in-class demonstration is required for this project. Not giving a demo to the class will result in a grade of 0.

CODING CONVENTION

- Each file shall have a file header.
- Each function shall have a function header.
- The code shall be properly indented and commented.
- The code shall include the lines in the *pgm_template.cpp* at the end of the *main()* function that will output your name and the compilation date, and pause the program.
- The program shall demonstrate modular design.
- Function recursion shall not be used.

PROJECT GRADING

A programming project is like a take-home test.

The student is given an extended amount of time, usually one or more weeks, in which to complete the program.

These items will be considered in grading the finished product:

1) Basics:
   (a) Is the project complete?
   (b) Is the code correctly compiled?
   (c) Does the program run by double-clicking?
   (d) Does the program run without crashing?
2) Did the student follow the written instructions?
   It is invalid to "reinterpret" or "change" the instructions in order to simplify the project.

SUBMISSION INSTRUCTIONS

Compile and test your code in the MinGW environment we have at TMCC. That is how it will be graded.

Create a folder named ***Proj2*** containing the following folders and contents:

Math_Testor, containing:
- *Math_Testor.exe*
- *Math_Testor.cpp*
- *.txt* file(s) (output files) (E.g., *firstname_lastname_results.txt*)

The ***screenshots.docx*** file shall be in the main folder: ***Proj2***.

The latest version should be compiled and running, and that is what will be tested for grading.
Zip up your folder and submit your ***Proj2.zip*** file to the Canvas Drop box.

Verification

To verify that your project is properly submitted, you can download your ZIP file from your Canvas.
Test your submission by performing the following procedure:

1. download your ZIP file from the Canvas drop box
2. unzip your ZIP file (extract its contents)
3. double-click on *Math_Testor.exe* to run it.
4. test the program to see if it behaves according to the specifications.

If the process (steps 1-4) doesn’t work for you, it won’t work for the grader. Fix it.