Assignment 7: refine the `mathq` program

It is time to "get your feet wet" with programming in Python. You will gain experience in modifying a Python program, finding your mistakes, debugging, and running it.

Follow the good programming practices that are posted in the *Programming Rules*, which includes:

- Placing a thorough, professional preamble at the beginning of the file
- Adding comments about your modifications to the original program
- Making sure your program is free of all errors when executed on any of the Linux Lab machines, prior to submission
- Acceptable style guidelines respective to the language and when distinguishing variables
- Avoiding error-prone syntax wherever possible

Instructions

Basically, you will modify the `mathq` program to make it satisfy a different set of requirements, and fix what is not working in it. Specifically, imagine that the client for whom we created the `mathq` program for would now like some changes. To be exact:

- Part 1: They want it to provide addition and subtraction problems as well as multiplication and division.
  - They still want the choice to be random, as before, but now the problem is randomly multiplication, division, addition, or subtraction.
  - For the addition and multiplication problems: the range of both operands is still limited to single digit numbers,
  - For the division and subtraction problems: The first operand is based on the second operand and the solution, where those are still limited to single digit numbers.
    So,

```
18 / 9  = ?
>
```

    is allowed, not `9 / 18 = ?` .

    Hence, the answers to the *division and subtraction* problems generated should be ***positive, single digit numbers*** in most cases.
    So,

```
18 / 9  = ?
>
```

    is allowed, not `1 - 2 = ?` .

- Part 2: Concerning division problems: The divide-by-zero problem should be fixed. Although zero can be an operand in multiplication (the first or the second), it cannot be the ***divisor*** (the second operand) in a division problem. So,

```
0 / 1 = ?
>
```

  is allowed, but not `1 / 0 = ?` .

- Part 3: Concerning subtraction problems: The ***subtrahend*** is the number that is subtracted from the ***minuend***. The problems cannot have value of zero as the minuend (the first operand), although it can be the subtrahend (the second operand).
  So,

```
6 - 0 = ?
>
```

  is allowed, but not `0 - 6 = ?` .

  The only situation where zero is permitted to be the minuend is it if is subtracted from itself. That is,

```
0 - 0 = ?
>
```

  is the only circumstance of zero minuend allowed.

- Part 4: The user should be able to type either the letter character 'q' or 'Q' to quit the `mathq` program.

- Part 5: When the program starts, in addition to displaying a "welcome message", it should display a message to the user about how to quit the program anytime.

- Part 6: Reporting a score would likely motivate the user to continue playing the game. The program should display the user's score after they quit the `mathq` program.

  - Each time the user answers a question correct, they earn 1 score point.
  - An incorrect answer is worth 0 points.
  - Only correct and incorrect answers are counted, not invalid responses.

  When the user quits, the program should display the total number of correct and incorrect responses and the percentage correct. For example, if the user answered 12 questions correctly and 6 questions incorrectly, the program would display something like:

```
You answered 12 out of 18 questions correctly, or 67% correctly.
Thank you for playing mathq.
```

The score is announced ***only*** when the user answers questions. When the user quits the `mathq` program immediately without properly playing the game, the score should not be announced, since they have not answered any questions in the appropriate manner to earn a score.

The expression of the user's score of correct answers as a percentage is not required for the score presentation.

All programs must be thoroughly tested before they are released to users. How should you test the program? This is an interactive program, so it is not a matter of creating many files and running the program on them. You have to pretend you are a user and play many games with it, giving it invalid responses, valid responses, right answers, wrong answers, and so on. Make sure that it counts the correct answers properly, and incorrect ones. What happens if the user quits without playing at all? Make sure it works as intended for that situation as well.

Grading Rubric

This assignment is graded on a 100 point scale.

The script will be graded on its correctness foremost. This means that it does exactly what the assignment states it must do, in detail. Correctness is worth 70% of the grade. Then it is graded on its clarity, simplicity, and efficiency, worth 30% of the grade.

The objectives when writing any script, are

- *clarity* – it should be easy to understand by someone with a basic knowledge of Python
- *efficiency* – it should use the least resources possible
- *simplicity* – it should be as simple as possible

Submitting Requirements

- **Due date:** Sunday November 1, 11:59PM Eastern Standard Time
- Late submissions will be docked total points at the rate of 1 point for every day it is late. No submissions will be accepted after Sunday November 8, 11:59PM Eastern Standard Time.
- Accepted formats:
    - Screenshot image(s) in JPEG, PNG, or TIF of the script, its execution for a sampling various input and the resulting output of each execution.
    - A zip archive file containing the modifed `mathq` script.
      The steps to create a zip archive and to secure-copy files is outlined in the tutorials called '`zip` command for beginners' and '`scp` command for beginners', that are located in the Course Materials section on the course Blackboard.