CPSC 1301, Computer Science I Programming Exercise 07

Week 7, MathGames.py

1 Introduction

This activity consists of four programming exercises. The following exercises are open book and open note. You are free to use any written documentation you wish. However, these are individual exercises, and you cannot consult with each other in writing your programs. Name your program `MathGames_lastname.py`.

This programming exercise has four parts consisting of four requirements. The grade for each requirement is indicated, for a maximum of 100 points. At a minimum, your program must compile successfully and run.

2 Exercise requirements

You will implement eight functions, as follows. In `main()`, you will call the functions to run the program.

interface() This method takes no parameters, prints the user interface to the console, prompts the user to specify the kind of problems to answer, i.e., addition, subtraction, multiplication, division, or exit. The method must also handle wrong input by the user.

numques() This method takes no parameters, prompts the user to specify the number of problems to answer (n) and the numeric range of the prolems (from 0 to *top*) and returns a two item *tuple*, `(n, top)`.

showMenu() This method takes no parameters, enters an infinite loop, calls `interface()` to get the two item tuple returned by that method, prompts tbe user to choose the type of problems to answer (addition, subtraction, multiplication, division, or exit), and calls the appropriate methods. This method will also accept a numeric grade from the arithmetic methods and report the grade. This method has no return value.

add((n,top)) This method takes a two item tuple consisting of the number of questions to answer (n) and the top number of the range (*top*), creates n random addition problems from 0 to n, records the number of correct answers, and returns the number correct as a floating point number when all the problems have been answered. It will also tell the student the correct answer to any problem the student misses.

sub((n,top)) This method takes a two item tuple consisting of the number of questions to answer (n) and the top number of the range (*top*), creates n random subtraction problems from 0 to n, records the number of correct answers, and returns the number correct as a floating point number when all the problems have been answered. It will also tell the student the correct answer to any problem the student misses. *Special requirement: Do not allow negative results.* If the *lhs* is less than the *rhs*, swap the values.

mul((n,top)) This method takes a two item tuple consisting of the number of questions to answer (n) and the top number of the range (*top*), creates n random multiplication problems from 0 to n, records the number of correct answers, and returns the number correct as a floating point number when all the problems have been answered. It will also tell the student the correct answer to any problem the student misses.

div((n,top)) This method takes a two item tuple consisting of the number of questions to answer (n) and the top number of the range (*top*), creates n random division problems from 0 to n, records the number

of correct answers, and returns the number correct as a floating point number when all the problems have been answered. It will also tell the student the correct answer to any problem the student misses. *Special requirement 1: Do not allow division by zero.* If the *denominator* is zero, generate new random values. *Special requirement 2: The student's answer will most likely be a floating point number.* You will have to find a way to accept a reasonable answer, given that $\frac{2}{3} = 0.6666667$, but you will consider either .66 or .67 correct.

get_rand_nums(top) This method will accept one parameter representing the top number of the range of random numbers to generate, and return a *tuple* of two numbers representing *lhs* and *rhs*. For example, if the student chooses 12 as the top of the range, the random number geneerator will select any integer from 0 to 12 inclusive.

3 Starter template

```
1  #!python
2  # Name: MathGames.py
3  # Author: Your Name
4  # Date: current date
5  # Purpose: Math Games, addition, subtraction, multiplication, division
6
7  import random
8  import math
9  from os import system
10
11 def hello():
12     print("Hello from 'MathGames.py'")
13
14 def showMenu():
15     print("Welcome to Math  Games\nPlease choose your test:")
16     pass
17
18 def interface():
19     pass
20
21 def numques():
22    pass
23
24 def get_rand_nums(top):
25     pass
26
27 def add(ntup):
28     pass
29
30 def sub(ntup):
31     pass
32
33 def mul(ntup):
34     pass
35
36 def div(ntup):
37     pass
38
39 #main function executes the defined functions
40 if __name__ == '__main__':
41     hello()
42     showMenu()
```

4 Sample output

```
Hello from 'MathGames.py'
Welcome to Math  Games
```

```
Please choose your test:

1. Addition
2. Subtraction
3. Multiplication
4. Division
9. Exit
        3
How many questions to you want to answer? 4
Enter the range of the test from 0 to ? 10
Multiplication
1. What is 10 * 10?
100
Correct
2. What is 3 * 1?
3
Correct
3. What is 8 * 0?
0
Correct
4. What is 9 * 6?
63
Incorrect, the answer is 54
You made a  75.0

1. Addition
2. Subtraction
3. Multiplication
4. Division
9. Exit
        8

Sorry, I don't understand

1. Addition
2. Subtraction
3. Multiplication
4. Division
9. Exit
        9
Goodbye
```