COP 2930 - Pair Programming Assignment #2

Due date: Please consult WebCourses for your due date/time

Directions: Please pick one person in the class to work with and sign up as a group in the group set PairProg2. This process will be covered in class.

Objectives

1. To give students practice working together to solve a problem.
2. To learn how to break a task into separate repeatable subtasks
3. To learn how to use functions to simplify program design.
4. To practice creativity via the Python Turtle.

Please choose either Problem A or Problem B to submit.

Problem A: UCF Flag (ucfflag.py)

Use the Python Turtle to design a new UCF Flag. In order to receive full credit, your design has to be reasonably complex and your code must use at least five functions total, one of which is a function main, which coordinates the whole drawing.

Please take a look at the in class example with polygons as to what I expect typical functions to do.

Thus, for your drawing, determine which pieces are repeatable subtasks. Then, determine what information your function needs to perform its subtask. Using this information, create the function prototype, clearly specifying in the comment what each formal parameter means. Then, write your function and test it individually, making sure it does what you want it do, before moving on.

You should do this for each subtask that maps to a function.

Then, use your functions in writing function main, which will coordinate the appropriate calls to the other functions to get the whole flag drawn.

If you prefer to draw something other than a flag, that is fine. Whatever you draw must be UCF themed in some way. Make sure you take a look at how to fill shapes via the Python Turtle.

Input Specification

There will be no input for this program.

Output Specification

The output is your turtle drawing!

Implementation Requirements

You use at least five functions, each of which carry out some clearly specified subtask.

Each function has a comment describing what each formal parameter represents as well as what the function does.

Each function has internal comments as well.

Problem B: Arithmetic Training, with functions (arithmetic.py)

In Individual Program #3, you wrote a program that gave the user 10 multiplication problems where both operands were in between 1 and 10, inclusive. For this program, we'll expand on this idea. Allow the user to play one of the following four games:

1) Addition Game
2) Subtraction Game
3) Multiplication Game
4) Division Game

At the beginning of the program, you'll give the user the following menu:

1) Addition Game
2) Subtraction Game
3) Multiplication Game
4) Division Game
5) Quit

If the user selects a number that isn't in between 1 and 5, tell them their choice was invalid and reprompt the to reenter their choice. Continue to do so until they enter something valid.

Once a valid choice is entered, execute that choice. So, for any of the games, just give the user 10 questions of that type. (For division, randomly generate two integers, a and b, in between 1 and 10, inclusive, and then ask the user what a*b divided by a is.) Please keep track of how many questions the user gets correct. At the end of the 10 questions, report how many questions out of 10 the user got for the round.

When the user enters quit, then show their overall statistics over all games played:

Total Questions Correctly Answers
Total Questions Posed
Percentage of Questions Answered Correctly

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

The user will always enter an integer for every query (menu choice and answer to a problem).

Output Specification
Largely, you may determine this, but here are a few guidelines:

Within each of the four games, please make sure to provide all of the information that was provided in the multiplication game for Individual Program #3.

When the user quits, please print out a summary of the following format:

```
You correctly solved X out of Y arithmetic problems, for a P% success rate.
```

where X is the number of problems correctly solved over all rounds, Y is the total number of problems posed over all rounds and P is the corresponding percentage. You may print P to any number of decimal places.

Note: You may add any enhancements you like, but please document them clearly and make sure I can easily see that you implemented the basic requirements for grading purposes.

Restrictions
Please IDLE 3.6 (or higher) to develop your program. Please submit EITHER **ucfflag.py** OR **arithmetic.py**

Your team's program should include a header comment with the following information: both group members' names, course number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details
Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility to IDLE.