CS Scholars Programming Hw2 - Written
Due Date: Friday 07/16 EOD

Name:

AndrewID:

For full credit on the assignment, complete either all Review + Core problems (#1-#10) or all Core + Spicy problems (#1-#6, #10-#12).

Bonus problems are related to the Advanced Track content, and are optional.

Core Problems - Written [60pts]

#1 - Variable Scope - 10pts

Can attempt after Function Definitions lecture

Consider the following code:

```
import random
import tkinter

def randomX(width):
    x = random.randint(0, width)
    return x

def randomY(height):
    return random.randint(0, height)

def getRandomScale():
    scale = random.randint(1, 5)
    return scale

def drawRandomImage(canvas, img):
    canvas.create_image(randomX(400),
                        randomY(400),
                        image=img)

# ignore graphics setup code here
root = tkinter.Tk()
canvas = tkinter.Canvas(root,
                        height=400,
                        width=400)
canvas.pack()

f = input("Type in an image filename:")
image = tkinter.PhotoImage(file=f)
scaleUp = getRandomScale()
image = image.zoom(scaleUp)
drawRandomImage(canvas, image)

root.mainloop() # ignore this line too
```

For the following two questions, ignore the graphics set-up code (root, canvas, etc).

Are there any **global** variables in this code? If yes, what are they?

Are there any **local** variables in this code? If yes, what are they, and which functions are they local to?

#2 - Function Call Stack - 5pts

Can attempt after Function Definitions lecture

Consider the following code:

```
def a(x):
    x = x + 2
    y = 2 * x
    print("b:", b(y))
    x = x / 2
    return x

def b(x):
    tmp = x - 2
    x = "wow"
    # PAUSE HERE
    return tmp

tmp = 10
result = a(tmp)
```

What will be on the function call stack when the code reaches the line `# PAUSE HERE` ?
You may not need to fill all the lines.

Call Stack

#3 - Evaluating Boolean Expressions - 10pts

Can attempt after Conditionals lecture

For each of the following Boolean expressions, determine whether it evaluates to `True`, `False`, or an error.

`(4 > 5) and ("foo" == "foo")`

- ☐ `True`
- ☐ `False`
- ☐ Error

`(10 > 0) or (0 == 1/0)`

- ☐ `True`
- ☐ `False`
- ☐ Error

`not (True and False)`

- ☐ `True`
- ☐ `False`
- ☐ Error

`(2 <= 5) and (4 + "a" == "4a")`

- ☐ `True`
- ☐ `False`
- ☐ Error

`("a" == "A") or (0 > 1)`

- ☐ `True`
- ☐ `False`
- ☐ Error

#4 - Code Tracing Conditionals - 15pts

Can attempt after Conditionals lecture

Given the following block of code, choose specific values for x, y, and z that would lead to the code printing A, B, C, D, or E. If one of the variables could be assigned to any value to achieve the result, write the word **"anything"** instead of a value. Fill out your answers in the table below.

```
if x < 10:
    if y > 20:
        if z == "foo":
            print("A")
    else:
        if y % 2 == 0:
            print("B")
        else:
            print("C")
elif x < 100:
    if y < 0 and z == "bar":
        print("D")
    elif y < 0:
        print("E")
```

Printed Result	x value	y value	z value
A			
B			
C			
D			
E			

#5 - Python Error Identification - 10pts

Can attempt after Testing and Debugging lecture

For each of the following lines of code, select whether it causes a **Syntax Error, Runtime Error**, or **No Error**. You are guaranteed that no code has a logical error.

```
print("Hello World"
```

- ☐ Syntax Error
- ☐ Runtime Error
- ☐ No Error

```
print(Test)
```

- ☐ Syntax Error
- ☐ Runtime Error
- ☐ No Error

```
print("2+2=" + 4)
```

- ☐ Syntax Error
- ☐ Runtime Error
- ☐ No Error

```
x - y = 5
```

- ☐ Syntax Error
- ☐ Runtime Error
- ☐ No Error

```
x = 1 == 2
```

- ☐ Syntax Error
- ☐ Runtime Error
- ☐ No Error

#6 - Writing Test Cases - 10pts

Can attempt after Testing and Debugging lecture

Assume we have written a program `isPositiveEvenInt(value)`, which returns `True` if the given value is a positive **and** even **and** an integer, and `False` otherwise.

Write a set of test case assertions for this function that fulfill the five test case types we discussed in lecture. Your test cases should be runnable in Python (use `assert` statements!)

Programming Problems

Each of these problems should be solved in the starter file available on the course website. They should be submitted to the Gradescope assignment Hw2 - Programming to be autograded. Make sure to check the autograder feedback after you submit!

For each of these problems (unless otherwise specified), write the needed code directly in the Python file **in the function definition** associated with the problem.

To test your code before submitting, 'Run file as script' and make sure all the test functions pass.

Review Problems - Programming [30pts]

#7 - `slope` - 10pts

Can attempt after Function Definitions lecture

Write a function that implements the following algorithm, which finds the slope of a line between two points. We have not included the function definition header - you'll need to add that in yourself under the problem's comment.

- The function's name is **slope**
- The function takes four arguments: **`x1`**, **`y1`**, **`x2`**, and **`y2`**.
- The function should follow this algorithm:
    a. First, compute the difference in y values (using subtraction) and assign it to the variable **`diffY`**.
    b. Second, compute the difference in x values and assign it to the variable **`diffX`**.
    c. Third, compute the slope (**`diffY`** divided by **`diffX`**) and assign it to the variable **`m`**.
- The function should return the variable **`m`**.

#8 - `numSign` - 10pts

Can attempt after Conditionals lecture

Write a function **numSign(x)** that takes a number as a parameter and returns a string representing its sign. More specifically, the function should return `"positive"` if the number is positive, `"negative"` if it is negative, and `"zero"` otherwise.

For example, `numSign(12)` should return `"positive"`, `numSign(-0.5)` should return `"negative"`, and `numSign(0)` should return `"zero"`.

You are guaranteed that the function will only be called on ints and floats.

#9 - Simple Debugging - 10pts

Can attempt after Testing and Debugging lecture

There are three functions in the Review part of the starter file that have been commented out with a triple-quote. Each function has exactly one bug. Use the debugging tactics we discussed in class to identify the bugs and fix them, then write a comment above each function explaining what the type of the bug was and how you fixed it. **Do not attempt to write the functions from scratch**; try to change as little as possible while fixing them.

The first function, `isEven(x)`, is supposed to return `True` if `x` is even, and `False` otherwise. For example, `isEven(12)` should return `True`.

The second function, `battleTextGenerator(person1, person2)`, is supposed to return a string with a battle announcement between the two listed people. For example, `battleTextGenerator("The Rock", "Hulk Hogan")` would return `"TONIGHT: The Rock VS Hulk Hogan"`.

The third function, `makeAdditionString(x, y)`, is supposed to take two integers and return a string showing the addition equation between the two. For example, `makeAdditionString(3, 4)` would return `"3 + 4 = 7"`.

Core Problems - Programming [10pts]

#10 - Interactive Program - 10pts

Can attempt after Conditionals lecture

In the function **`interactiveProgram`,** use the `input` function and conditionals to set up a short interactive program of your own design. This could be a very short choose-your-own-adventure story, or a quiz, or whatever else you'd like! The only requirements are:

1. You must use the `input` function to collect information from the user at least three times, and at least one of the `input` results must be cast to a different data type.
2. You must use **conditionals** somewhere in your code. There should be at least one `if` statement and at least one `elif` or `else` statement.
3. All the code for your interactive program must be in the `interactiveProgram` function (or helper functions that `interactiveProgram` calls).

Spicy Problems - Programming [30pts]

#11 - Three Lines Area - 20pts

Can attempt after Function Definitions lecture

For this problem, you will implement **four functions**. One, **`threeLinesArea(m1, b1, m2, b2, m3, b3)`**, will be the primary function; its job is to take the slopes and intercepts of three lines and calculate the area between them. The other three functions support this function in its work.

The first helper function, **`lineIntersection(m1, b1, m2, b2)`**, takes the slopes and intercepts of two lines and returns the x value of the point where they intersect. Recall that this can be calculated as:

$$\frac{b2 - b1}{m1 - m2}$$

The second helper function, **`distance(x1, y1, x2, y2)`**, takes two (x, y) points and returns the distance between them. Recall that this can be calculated as:

$$\sqrt{(x2^2 - x1^2) + (y2^2 - y1^2)}$$

The third helper function, **`triangleArea(a, b, c)`** takes three numbers (lengths of the sides of a triangle) and returns the area of a triangle with those sides. This can be calculated with Heron's formula as:

$$\sqrt{s(s - a)(s - b)(s - c)}$$

Where **s** is the semi-perimeter of the triangle, calculated as:

$$\frac{a + b + c}{2}$$

Once you've implemented the three helper functions, use them in `threeLinesArea` to find the points where each pair of lines intersects, then return the area of the triangle formed by connecting these three points. You may assume that the lines definitely form a triangle (the lines are not parallel).

#12 - Advanced Debugging - 10pts

Can attempt after Testing and Debugging lecture

There is a function in the Spicy part of the starter that has been commented out with a triple-quote. This function has exactly three bugs. Use the debugging tactics we discussed in class to identify the bugs and fix them, then write a comment above the function explaining what types of the three bugs were and how you fixed them. **Do not attempt to write the function from scratch**; try to change as little as possible while fixing it.

The function should return `True` if the input `x` is either an even integer less than or equal to 10, or an odd integer greater than 10. The function should return `False` otherwise.

Bonus Problems [10pts]

Advanced Programming 1 - Language Types - 2.5pts

Choose a programming language that is **not** one of the following: Python, Java, HTML/CSS, Javascript. This should preferably be a language you have not used before.

You can find a list of programming languages here:
https://en.wikipedia.org/wiki/List_of_programming_languages

What language did you choose?

Is your language mostly imperative, declarative, both, or neither?

Which programming paradigm best describes your language?

Describe a variation (like weak/strong typing, etc.) that makes this language interesting.

Advanced Programming 2 - Language Types - 2.5pts

The slides described two non-procedural programming paradigms that Python supports - **object-oriented programming** and **functional programming**. Choose one of these paradigms and write a short Python program in that style. Don't just rewrite the Coordinate/distance example from the slides- create something new!

If you choose to create an object-oriented program, you must have:

- At least one class
- At least one instance of that class
- At least two properties in the class
- At least one method in the class (apart from __init__)

If you choose to create a functional program, you must have:

- At least one lambda function
- At least one call to that lambda function
- No variables that change state during a program run (it's okay to set a variable once if it doesn't change after that)

You should complete this work in a separate file from hw2.py, and upload that file to Hw2 - Bonus, so that you don't break the autograder.

Advanced Computer Science - The Internet - 5pts

Select all of the steps that might occur when you click on a website link in your browser. (Steps are not listed in order, and may not all happen in the same request).

- ☐ A packet is sent to the DNS server to ask for the IP address of the website
- ☐ A packet is sent to the DNS server to ask what your computer's IP address is
- ☐ The route your packet request takes through routers today is different from the route it took to get to the same server yesterday
- ☐ The route your packet request takes through routers today is the same as the route it took to get to the same server yesterday
- ☐ The server sends the website back to your computer as a bunch of small packets
- ☐ The server sends the website back to your computer via routers as one large packet
- ☐ The website sent back by the server is sent through the cloud to allow for faster transfer
- ☐ Your browser shows you part of the website content while it waits for the remaining packets to arrive
- ☐ Your ISP gets a bunch of packets that represent the website and reassembles them into one large packet, which it sends to your computer
- ☐ Your ISP intercepts the request for an IP address based on a URL and replies with the address you're looking for, which it had cached