CS Scholars - Programming Hw2 - Written
Due Date: Friday 07/15 EOD

Name:

AndrewID:

For full credit on the assignment, complete either all Review + Core problems (#1-#10) or all Core + Spicy problems (#1-#7, #10-#12).

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

Core Problems - Written

#1 - Variable Scope

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()
# stop ignoring now

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 Tracing

Can attempt after Function Definitions lecture

Consider the following code:

```
1  def fb(x):
2    tmp = x + 2
3    x = "wow"
4    print("a:", x)
5    return tmp
6
7  def fa(x):
8    x = x + 3
9    y = 4 * x
10   print("b:", fb(y))
11   x = x / 2
12   return x
13
14 tmp = 5
15 result = fa(tmp)
16 print("Done!")
```

First, what will this block of code print once it has finished executing? **Enter your answer in the space below.**

List all the function calls that occur in this code block with their **name, argument value(s),** and **returned value**. If there is no name / argument / returned value, leave the space blank. You might not need to use the whole table.

Note: Do not include any calls to built-in functions in the table.

Function name	Argument value(s)	Returned value

#3 - Evaluating Boolean Expressions

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

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

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 and that no variables are defined before the code runs.

```
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 - Loop Control Variables

Can attempt after Loops lecture

Each of the following problem prompts could be implemented using a loop. Identify the start value, continuing condition, and update action for the loop control variable you would use in that loop. Assume that the loop control variable will be outputted at the beginning of the loop, and no conditional will be used. We've given an example of what this looks like in the first line

Ex) Output the numbers from 1 to 10, inclusive.

A) Output all even numbers between 2 and 20, exclusive on 20 (but not 2).

B) Output the numbers from 10 to 1, inclusive on both.

C) Output the numbers 3, 9, 15, 21.

Prompt	Start Value	Continuing Condition	Update Action
Ex	1	x <= 10	x = x + 1
A			
B			
C			

#7 - While Loops

Can attempt after Loops lecture

Given the following block of code, fill out a variable table that shows the values of the variables at the **end** of each iteration of the loop. You may not need to fill out values for every listed iteration.

```
x = 0
y = 10
z = 0
while x <= y:
    x = x + 3
    y = y + 1
    z = (x + y) - z
    print(x, y, z)
```

	x value	y value	z value
Pre-loop	0	10	0
Iter 1			
Iter 2			
Iter 3			
Iter 4			
Iter 5			
Iter 6			
Iter 7			
Iter 8			

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

#8 - `slope`

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`**.

#9 - `printSquare`

Can attempt after Loops lecture

Write a function `printSquare(n)` which prints an ascii art square out of asterisks based on the integer `n`. For example, `printSquare(5)` would print the following:

```
*****
*****
*****
*****
*****
```

Note that the square is five lines long, with each line having five asterisks. As another example, `printSquare(8)` would look like:

```
********
********
********
********
********
********
********
********
```

You'll want to create a loop where each iteration prints a single line of the square. There's a **string operator** that will help you draw multiple stars on each line.

Note: `n` is guaranteed to be positive.

Core Problems - Programming

#10 - Interactive Program

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 Buzzfeed-style 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.
2. The `interactiveProgram` function must take no parameters
3. You must use **conditionals** somewhere in your code. There should be at least two `if` statements and at least one `elif` or `else` statement.
4. All the code for your interactive program must be in the `interactiveProgram` function (or helper functions that `interactiveProgram` calls).

Spicy Problems - Programming

#11 - nthFibonacciNumber

Can attempt after Function Definitions lecture

Write the function `nthFibonacciNumber` which takes an integer `n` and returns the nth number in the Fibonacci sequence. For example, `nthFibonacciNumber(10)` would return `55`, as the first ten numbers in the Fibonacci sequence are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55. (Note that a function header is not provided - you'll need to create it yourself).

How can you compute this number? You could use the definition of Fibonacci numbers, which states that fib(n) = fib(n-1) + fib(n-2). But there's an easier way! The nth Fibonacci number can be calculated in a closed-form expression by using the golden ratio, φ.

$$fib(n) = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

Unfortunately, Python doesn't have φ built in. But you can compute the number yourself with a basic equation:

$$\varphi = \frac{1+\sqrt{5}}{2}$$

Note that Fibonacci numbers are all **integers**, and this expression will indeed produce a number very close to an integer, but it may sometimes be a bit off due to inaccuracies in Python calculations. Make sure to **round** the result before returning it to get the answer as an integer.

#12 - `printTriangle`

Can attempt after For Loops lecture

Write a function `printTriangle(n)` which prints an ascii art triangle out of asterisks based on the integer `n`. For example, `printTriangle(5)` would print the following:

```
*
**
***
**
*
```

Note that the triangle is five lines long, with the top and bottom line each having only one asterisk, the second and second-from bottom lines each having two asterisks, etc. So `printTriangle(9)` would look like:

```
*
**
***
****
*****
****
***
**
*
```

You'll want to create a loop where each iteration prints a single line of the triangle. Note that `n` is guaranteed to be positive and odd.

Hint: how can the program switch from increasing to decreasing? Consider using two separate loops (one going up, one going down). You can also do this with a conditional inside the loop - we'll talk more about how to do that next week.

Bonus Problems

Advanced Programming 1 - Recursion

Assume you want to write a function `recursiveSum` that takes a positive integer, `n`, and **recursively** computes the sum from one to `n`.

For example, the result when calling the function on n=5 is 5+4+3+2+1 = 15.

What condition do you need to check for your **base case**, and what do you return?

What is the recursive call on a smaller problem in the **recursive case**, and how do you use that result to solve the whole problem for n?

Advanced Programming 2 - Recursion

In the programming starter file, write the function `powerSum(n, k)` that takes two non-negative integers `n` and `k` and returns the so-called power sum: $1^k + 2^k + ... + n^k$. You must use **recursion** to solve this problem: for loops, while loops, and the function `sum` are not allowed.

Note that the test function for `powerSum` is commented out; you'll need to uncomment it to test your function.

Advanced Computer Science - Concurrency

A factory with four workers produces custom t-shirts. To make a t-shirt, the workers follow these steps:

- [S] Set up supplies (for measuring) (5 minutes)
- [M] Measure the fabric (5 minutes)
- [S] Set up supplies (for cutting) (5 minutes)
- [C] Cut out the pattern (5 minutes)
- [S] Set up supplies (for sewing) (5 minutes)
- [W] Sew it all together (5 minutes)
- [F] Fold the shirt (5 minutes).

Note that setup occurs **once** before starting either measuring, cutting, or sewing. When you set up new supplies for a task, you put away the supplies for the previous task.

Continue to next page

Originally each worker made one shirt at a time, with all four workers working in parallel. Each of the cells in the following table represents five minutes, with the whole table representing an hour of work. Fill in the cells with the letters representing the steps to demonstrate the original system the factory used.

Worker	00:00	00:05	00:10	00:15	00:20	00:25	00:30	00:35	00:40	00:45	00:50	00:55
A												
B												
C												
D												

How many complete, folded shirts could be made by four workers in one hour with the original system?	

Recent budget changes have led to new restrictions on materials. Now the workers have to share a single sewing machine; in other words, only one worker can sew at any given point in time. They decide to use a new approach to make things more efficient.

Create a new schedule that uses **pipelining** to increase the efficiency of the shirt-making process. (**Hint:** think about the most efficient way to split up the tasks.)

Worker	00:00	00:05	00:10	00:15	00:20	00:25	00:30	00:35	00:40	00:45	00:50	00:55
A												
B												
C												
D												

How many complete, folded shirts could be made by four workers in one hour using the new pipeline?	