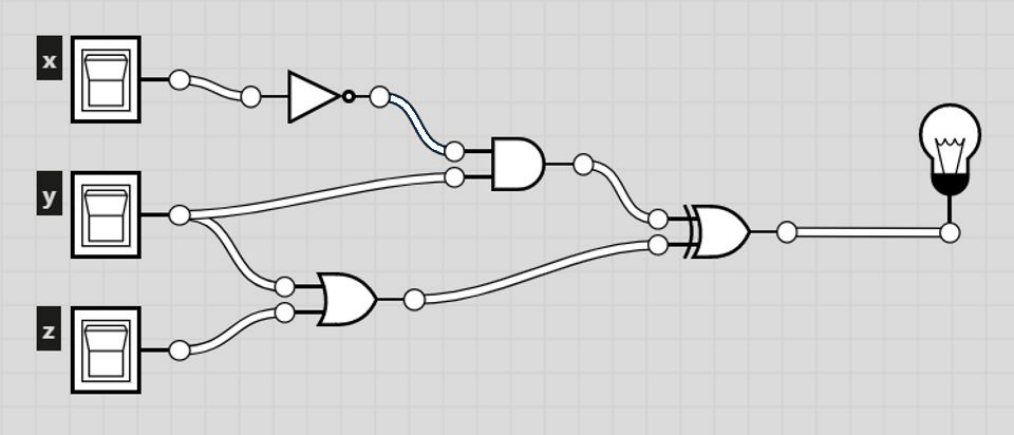
15-110 Check2 - Written Portion

Name:

AndrewID:

#1 - Circuit to Truth Table and Expression - 24pts

Fill out the truth table below to show all possible input combinations and all resulting outputs for the circuit shown here. You may not need to use all the rows.



x value	y value	z value	output value

Write a Boolean expression which is equivalent to the circuit above in the box below.
Note: it's fine to use normal Boolean terms (and/or/not/xor) instead of circuit operators.

#2 - Full Adder Facts - 9pts

In class and in the lecture slides, we showed how to put together a Full Adder circuit. For each of the following questions, choose the **best** answer as relates to that circuit.

What are X and Y?

- ☐ The two whole numbers being added
- ☐ Single binary digits of the two numbers being added
- ☐ Two binary digits of the first number being added

What is C_{in}?

- ☐ The third whole number being added
- ☐ A single binary digit of the third number being added
- ☐ The number carried in from the previous addition
- ☐ The remainder of the current addition

Why do we need two output values?

- ☐ To manage the large number of gates
- ☐ To account for both of the inputs
- ☐ To hold both the result and the original number
- ☐ To hold both the result and the number that will be carried over

#3 - Code Tracing While Loops - 18pts

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			

#4 - Code Tracing with For Loops - 9pts

For each of the following range expressions, list all the values the loop variable will be set to over the course of the range. For example, `range(1, 5)` produces `1`, `2`, `3`, `4`.

Range Expression	Numbers Produced
`range(3)`	
`range(4, 8)`	
`range(1, 10, 3)`	

15-110 Check2 - Programming Portion

Each of these problems should be solved in the starter file available on the course website. Submit your code to the Gradescope assignment Check2 - Programming for autograding.

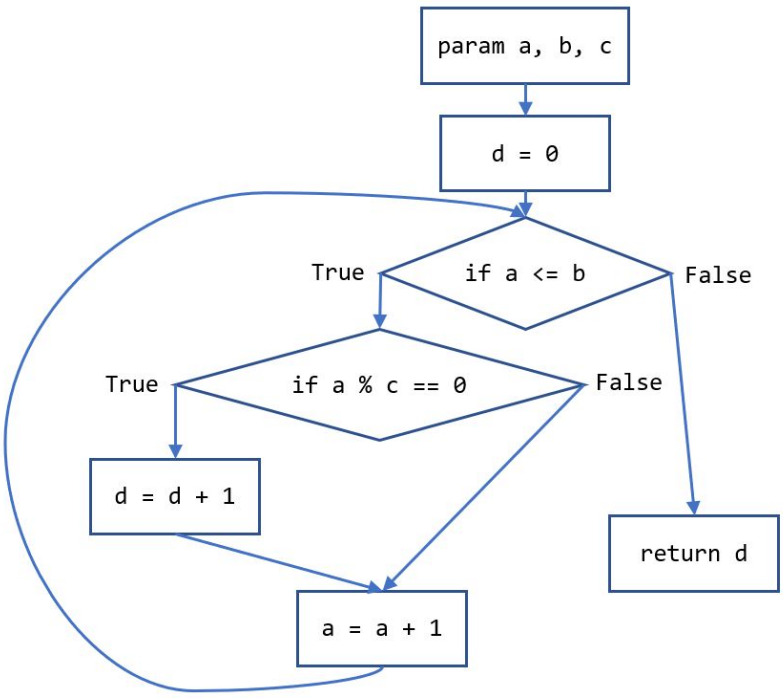
All programming problems may also be checked by running the starter file, which calls the function `testAll()` to run test cases on all programs.

Note: Check2 & Hw2 are the first assignments where you will need to do a substantial amount of coding. We encourage everyone to make good use of Piazza, office hours, and small group sessions to get help.

In particular, if you attend a small group session in Week 3 after Wednesday, your TA will include Problem #2 (`drawIllusion`) as one of the practice problems and will provide more help in solving the problem than is usually available at office hours.

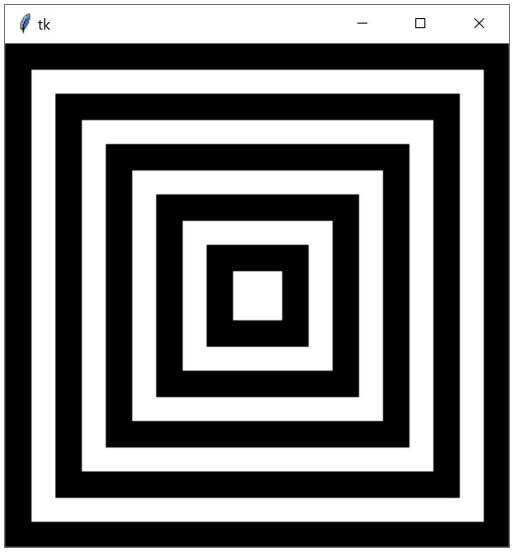
#1 - Flow Chart to Program - 10pts

Given the control flow chart below, write a function `mysteryFunction(a, b, c)` that implements the control flow chart correctly.



#2 - `drawIllusion(canvas)` - 20pts

Write the function `drawIllusion(canvas)` which takes a Tkinter canvas and draws the illusion shown below. You must use a **loop** to do this; don't hardcode a large number of rectangles.



Hint: it's easiest to make this illusion by drawing **overlapping squares**. Start with the largest black square, then draw the next-largest white square, etc. You'll need to draw 10 squares total. The canvas is 400px wide, so each square should be 20 pixels smaller on each side than the previous one (with the last square being exactly 40 pixels wide).

Another Hint: start by considering what the loop control variable should be. Which values need to change as you move to the next square? How do those values relate to the loop control variable? Consider our approach to drawing a grid in lecture as well.

#3 - `factorial(x)` - 10pts

Write the function `factorial(x)` which takes a non-negative integer, `x`, and returns x!. Recall that x! = x*(x-1)*(x-2)*...*3*2*1. You may not use the built-in function `math.factorial()`; that would make this too easy. Instead, you must use a **for loop** to solve this problem.

Hint: consider the sum-1-to-10 problem we went over in lecture. You can use a very similar approach to solve this problem.