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.

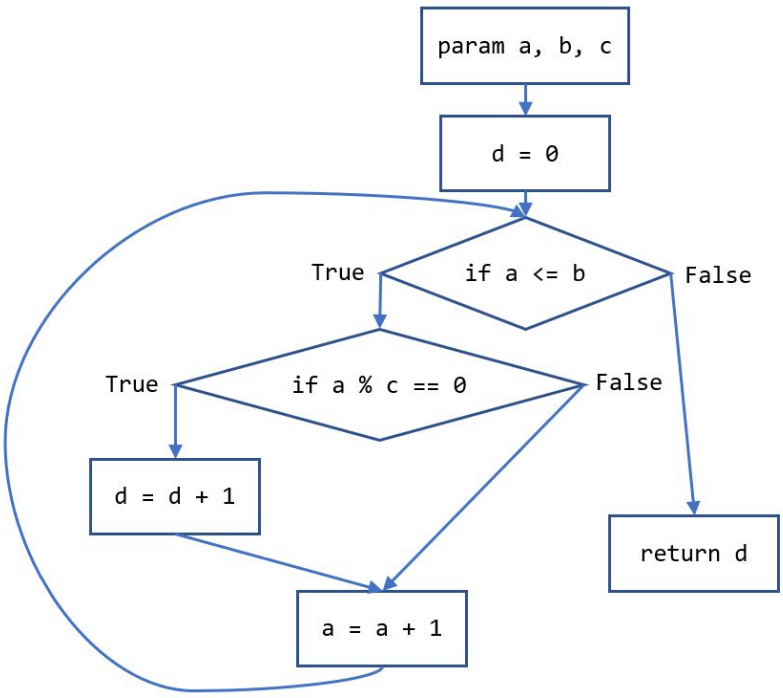
All programming problems may also be checked by running the starter file, which calls the function `testAll()` to run test cases on all programs.

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.

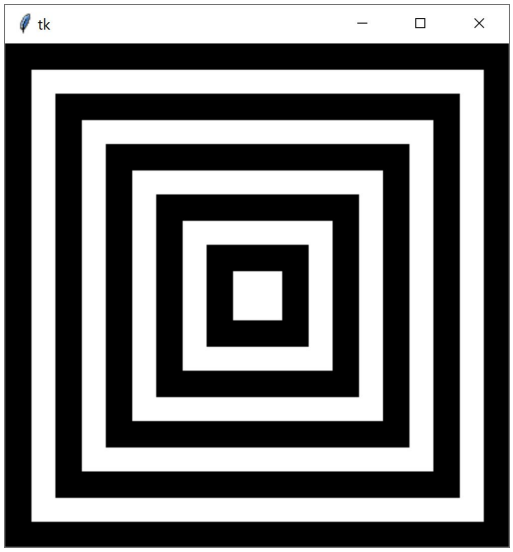
#1 - Flow Chart to Program - 10pts

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.