# Exam 5

*Comp 123*

- You may use the computer, and any resources on Moodle, Piazza, the Interactive Python textbook, your online journal, or the Python.org documentation pages. You may also use any paper resources you like.
- You **may and should** ask me for clarifications, or hints.
- You have 30 minutes to complete this test.

1. (10 pts) In the `exam5Code.py` file, you will find the beginnings of a GUI program. This program will play a "Knock Knock" joke with the user. If you haven't seen the format before, see Wikipedia's Knock Knock joke entry or, more fun, see examples.

   A typical Knock Knock joke involves two people, the joke teller and the responder. It has a relatively fixed pattern ending in a (bad) pun. See this example:

   Joker: Knock Knock
   Resp: Who's there?
   Joker: Canoe
   Resp: Canoe who?
   Joker: Canoe help me with my homework?

   This program will implement just one Knock Knock joke, one that was a favorite with my kids years ago because it violates the pattern of the joke, and makes the responder the unwilling victim. It goes like this:

   Joker: Knock Knock
   Victim: Who's there?
   Joker: Banana
   Victim: Banana who?
   Joker: Knock Knock
   Victim: Who's there?
   Joker: Banana
   Victim: Banana who?
   … (continues this cycle until Joker decides to end the pattern)
   Joker: Knock Knock
   Victim: Who's there?
   Joker: Orange
   Victim: Orange who?
   Joker: Orange you glad I didn't say Banana?

You'll make a GUI to play this joke, using a label to display the joker's next line, and a button for the user to click with the correct victim's response. The content of the label and the text of the button will change with every click of the button.

I have written the most complicated part for you: the logic that updates the joker label and victim button with the correct next text, and that chooses randomly when to end the cycle. All this logic is in the callback function for the button: `victimResponse`.

Your task is to write the `knockKnockGui` function, which takes no inputs and sets up and runs the GUI. Use the global variables I've provided. Your code should set up the main window, and add a label and a button to it. Be sure that:

a. The widgets are assigned to the global variables
b. You set the global variable `count` to be 1 inside your function
c. The text in the label at the start is `'Knock Knock'`
d. The text in the button is **exactly** `"Who's there?"` at the start
e. The button's command is set to my callback function, `victimResponse`
f. You call the main window's `mainloop` method at the end of the function

2. **You may do question 2 either online or on paper. I'll provide paper and pencils/pens as needed.** At the bottom of the exam5Code.py file, you will find a recursive function that performs a simple version of the Towers of Hanoi puzzle (see an explanation of the puzzle on Wikipedia and a playable version at Math Is Fun). The general idea is that you have three posts, A, B, and C, and some variable number of rings of different sizes stacked on post A. You want to move all the rings from post A to post C. You can only move one ring at a time, and must set it down on a post before moving another. You can set rings on top of other rings, but never a larger ring on top of a smaller one: only smaller on larger.

   The code I've given you recursively solves the Towers of Hanoi puzzle: it prints out the optimal sequence of moves you should make to solve the problem. For instance, is you call it with A as the start post, C as the goal, and B as the spare post, and two rings (assumed to be stacked on A at the start), then it will print:

   Move ring on A to B
   Move ring on A to C
   Move ring on B to C

   Your task is to draw a happy robot diagram to show all the calls that would result from the call below (including the call itself):

```
towers('A', 'C', 'B', 3)
```

You must show every call, and what all the input arguments are. Show which call makes which other call. There are no return values, so you don’t have to show that.