Homework Day 4 Instructions

Problem 1

Preamble: In this problem, you will implement a popular game in South Korea called the 3-6-9 (sam-yuk-gu) game. In this game, you play with a number of people (the more, the better!) sitting in a circle. Then, people in the group take turns counting to a number, starting from 1, 2 and so on. However, when the number has a '3', or '6', or '9', you do not say the number and clap instead. The number of times you clap will equal the number of times ' 3-6-9 ' occurs in that number. For example, you clap one time for '6' but clap twice for '33' and '39'. You will lose the game if you forget to clap correctly.

Instructions

Write a function lets_play_369(N) which receives the 'last number to be played'/total number of rounds as input N

- your function should count the number of claps happening in the game and return it as an output count_clap
- N is an integer less than 100
- you are not required to print out the game play record. However, it is highly recommended to visualize the counts while playing

```
count = lets_play_369(13)
# output:
# 1,
# 2,
# clap! (3),
# 4,
# 5,
# clap! (6),
# 7,
# 8,
# clap! (9),
# 10,
# 11,
# 12,
# clap! (13),
print(count)
# output: 4
```

Problem 2

Write a function `zodiac` that takes in a year and prints the person's zodiac animal, using this calendar.

HINT: Look at what the modulo value of each year is, and make a dictionary using those possibilities!

```
year = 2023
ans = zodiac(year)
print(ans)
# result: "rabbit"
```

Problem 3

Write a function `fib` which takes in a number `L` (which must be at least 1) and prints out the fibonacci sequence of that many numbers

you will need to track how many numbers you have printed, as well as the values of the numbers you are working with in the sequence!

optional challenge: can you make the code print the sequence in one line?

```
fib(3)
# will print 1, 1, 2
```

Problem 4

In this problem, you will write a function get_to_know which takes one argument N, which should be a number between 2 and 5. Then, you will ask your user N many questions to get to know them better!

The function should ask the user questions of your choice (for example, their name, age, and favorite food in the dining halls). At least one answer should be a string and at least one should be a float or int. As you learn about them, add their answers to a dictionary as values, with keys you choose.

After asking all of these questions, you should print “nice to meet you!” and a message including at least one answer to a question you asked. The function should return the dictionary with your information.

1. Hint1: you can make an empty dictionary by just setting empty curly brackets! d = {}.
2. Hint2: you will need to use a for loop!
3. Hint3: You can put your questions into a dictionary, with keys as the numbers of the questions.

```
get_to_know(2)
# whats your name? ratty rat
# whats your age? 900001
# it was great to meet you, ratty rat!
# {'q1': 'ratty rat', 'q2': 900001}
get_to_know(5)
# whats your name? blueno
# whats your age? 1
# whats your favorite food? blueberries
# do you like brown? y/n y
# have you seen the movie totoro? yes!!
# it was great to meet you, blueno!
# {'q1': 'blueno', 'q2': 1, 'q3': 'blueberries', 'q4': 'y', 'q5': '
```

Problem 5 -- Optional

This problem explores a concept called *recursion* which we do not have time to cover in class. Recursion is when a function calls itself, so that multiple copies of it run until some value / option is reached. An example is included below, which you can follow with environment diagrams. Try to see what this function does (it is a computation you know!).

```
def mystery(a, b):
  if b == 1:
    c = a
  else:
    c = a + mystery(a, b - 1)
  return c
```