CS-UY 1114: Lab 4

`for` Loops and `while` Loops

You must get checked out by your lab CA **prior to leaving early.** If you leave without being checked out, you will receive 0 credits for the lab.

Restrictions

The Python structures that you use in this lab should be restricted to those you have learned in lecture so far. Please check with your teaching assistants in case you are unsure whether something is or is not allowed!

Create a new python file for each of the following problems.

Your files should be named *lab[num]_q[num].py* **similar to homework naming conventions.**

Problem 1: *Loops Galore!*

Solve this problem by hand.

What would be printed for each of the following code snippets? If there is an infinite loop state that.

(a)

```
for i in range(0,10):
    print(i)
```

(b)

```
for i in range(1, 13, 2):
    print(i)
```

(c)

```
for i in range(43, 31, -3):
    print(i)
```

(d)

```
num = 1
while num < 28:
```

```
    print(num)
    num *= 3
```

(e)

```
num = 13
while num < 23:
    if num % 2 == 0:
        print(num)
    else:
        print("fizz")
    num += 1
```

Problem 2: *Multiple Use Calculator*

This problem is a brief extension on last weeks single use calculator **(Lab 3 Problem 4)**. This extension will let the user use the calculator as many times as they want.

1. Begin by copying your solution from the *Single Use Calculator* into a new file.
2. Now make the additions necessary to have the calculator run until the user inputs Q to quit. Otherwise they should just hit enter to continue making calculations.

The following is the expected output of the program:

```
This is a four operation calculator.
Hit enter to continue and Q to quit calculator:
Enter your first number: 2
Enter the operation (+, -, *, /): +
Enter your second number: 3
2.0 + 3.0 = 5.0
Hit enter to continue and Q to quit calculator:
Enter your first number: 3
Enter the operation (+, -, *, /): *
Enter your second number: 9
3.0 * 9.0 = 27.0
Hit enter to continue and Q to quit calculator:
Enter your first number: 3
Enter the operation (+, -, *, /): 2
Enter your second number: 3
3.0 2 3.0 is an invalid operation.
Hit enter to continue and Q to quit calculator: Q
Goodbye!
```

Problem 3: *Fibbonacci*

The Fibonacci sequence is a sequence in which each number of the sequence is the sum of the previous two numbers. The first two numbers of the sequence are defined as 1 and 1. Observe for the following first ten

numbers of the sequence that the sum of the previous two numbers gives the next number:

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

For example, `1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5` and so on.

Create a new Python file and write code that takes a positive integer, n, and prints the first n terms of the Fibonacci sequence. For example, a sample code execution shown below is:

```
Please enter a number: 7
1
1
2
3
5
8
13
```

Restriction: You may not use recursion.

Problem 4: *I've Got the Power(s)*

This problem will be printing the powers of a given base and every number up to the maximum power.

1. Ask the user for two **positive integers**, a `base`, and a `power`, and
2. Print, one by one, the result of raising that base by every power from 0 to `power`.

Here's an example of this programs execution:

If, for example, the user enters `2` and `7`:

```
Please enter a positive integer to serve as the base: 2
Please enter a positive integer to serve as the highest power: 7
2 ^ 0 = 1
2 ^ 1 = 2
2 ^ 2 = 4
2 ^ 3 = 8
2 ^ 4 = 16
2 ^ 5 = 32
2 ^ 6 = 64
2 ^ 7 = 128
```

If the user instead enters `10` and `3`:

```
Please enter a positive integer to serve as the base: 10
Please enter a positive integer to serve as the highest power: 3
```

```
10 ^ 0 = 1
10 ^ 1 = 10
10 ^ 2 = 100
10 ^ 3 = 1000
```

A few things to keep in mind:

- You may assume the user input will be numerical.
- Since the options for the `base` and the highest `power` are literally infinite, the use of a **loop** will come in very handy.
- If the user attempts to enter a negative number into your program, **print an error message instead of the list of powers.**
- If the user attempts to enter a float into your program, **print an error message instead of the list of powers.** There's a few ways to check for this, but make sure it's something that you have learned in class already!

```
Please enter a positive integer to serve as the base: 3.4
Please enter a positive integer to serve as the highest power: 4
ERROR: Both values must be POSITIVE INTEGERS.
```

```
Please enter a positive integer to serve as the base: -345324325436
Please enter a positive integer to serve as the highest power: 2
ERROR: Both values must be POSITIVE INTEGERS.
```

Problem 5: *Higher and Higher*

Write a program, that will find the largest value out of X-number of user-entered **positive** values. This will be done in three steps:

1. Asking the user to enter the number of values they want to enter.
2. Asking the user to enter the values.
3. Printing the largest of these values (what happens if the user entered `0` in step 1?)

This behavior will look like this:

```
Please enter how many positive values you want to consider: 5
Enter your values:
24
27.5
30.234
100
1
The largest of these values is 100.0.
```

Do NOT use lists for this problem, if you know what lists are. Again, that's no fun.

Problem 6: Blackjack

Blackjack is a card game played between the player and a dealer where the goal is to get the sum of your cards as close to 21 as possible. We'll create a modified version of the game according to these rules:

- At the start of the round, the dealer will receive a random card sum within the range [2, 21]. The dealer cannot pick up more cards after this point
- At the start of the round, the player will receive a random card of value [1, 11]
- A player may `"DEAL"` to receive another random card of value [1, 11] or `"STAND"` to stay with the cards they currently have

The win conditions are as follows:

- The player can win *only if* the sum of their cards is less than or equal 21 and if their card sum is closer to 21 than the dealer's
- The player can tie if the sum of their cards is equal to the dealer's sum
- Otherwise, the player loses

Your program must:

1. Print a welcome message to the user
2. Each round, print the user's card sum and prompt the user for an action
3. Display to the user whether they won, lost, or tied

Here are some example outputs:

```
Welcome to Blackjack!
Your current card sum is: 6
What would you like to do next?: (DEAL, STAND) DEAL
Your current card sum is: 16
What would you like to do next?: (DEAL, STAND) STAND
You won! Your card sum was 16 and the dealer's was 15
```

```
Welcome to Blackjack!
Your current card sum is: 4
What would you like to do next?: (DEAL, STAND) DEAL
Your current card sum is: 12
What would you like to do next?: (DEAL, STAND) DEAL
Your current card sum is: 14
What would you like to do next?: (DEAL, STAND) DEAL
Your current card sum is: 23
What would you like to do next?: (DEAL, STAND) DEAL
You lost! Your card sum was 23 and the dealer's was 14
```

Think about what type of loop to use for this and how to structure the loop before you start coding