CS 61A: Homework 1

Due by 11:59pm on Wednesday, 1/28

Instructions

Download hw01.zip. Inside the archive, you will find a file called hw01.py, along with a copy of the OK autograder.

Submission: When you are done, submit with `python3 ok --submit`. You may submit more than once before the deadline; only the final submission will be scored.

Using OK

The `ok` program helps you test your code and track your progress. The first time you run the autograder, you will be asked to log in with your @berkeley.edu account using your web browser. Please do so. Each time you run ok, it will back up your work and progress on our servers. You can run all the doctests with the following command:

```
python3 ok
```

To test a specific question, use the `-q` option with the name of the function:

```
python3 ok -q <function>
```

By default, only tests that fail will appear. If you want to see how you did on all tests, you can use the `-v` option:

```
python3 ok -v
```

If you do not want to send your progress to our server or you have any problems logging in, add the `--local` flag to block all communication:

```
python3 ok --local
```

When you are ready to submit, run `ok` with the `--submit` option:

```
python3 ok --submit
```

Readings: You might find the following references useful:

- Section 1.2
- Section 1.3
- Section 1.4
- Section 1.5

Table of Contents

Question 1

We've seen that we can give new names to existing functions. Fill in the blanks in the following function definition for adding `a` to the absolute value of `b`, without calling `abs`.

```
from operator import add, sub

def a_plus_abs_b(a, b):
    """Return a+abs(b), but without calling abs.

    >>> a_plus_abs_b(2, 3)
    5
    >>> a_plus_abs_b(2, -3)
    5
    """
    if b < 0:
        f = _____
    else:
        f = _____
    return f(a, b)
```

Question 2

Write a function that takes three positive numbers and returns the sum of the squares of the two largest numbers. Use only a single expression for the body of the function:

```
def two_of_three(a, b, c):
    """Return x*x + y*y, where x and y are the two largest members of the
    positive numbers a, b, and c.

    >>> two_of_three(1, 2, 3)
    13
```

```
    >>> two_of_three(5, 3, 1)
    34
    >>> two_of_three(10, 2, 8)
    164
    >>> two_of_three(5, 5, 5)
    50
    """
    "*** YOUR CODE HERE ***"
```

Question 3

Let's try to write a function that does the same thing as an `if` statement.

```
def if_function(condition, true_result, false_result):
    """Return true_result if condition is a true value, and
    false_result otherwise.

    >>> if_function(True, 2, 3)
    2
    >>> if_function(False, 2, 3)
    3
    >>> if_function(3==2, 3+2, 3-2)
    1
    >>> if_function(3>2, 3+2, 3-2)
    5
    """
    if condition:
        return true_result
    else:
        return false_result
```

Despite the doctests above, this function actually does not do the same thing as an `if` statement in all cases. To prove this fact, write functions `c`, `t`, and `f` such that `with_if_statement` returns the number `1`, but `with_if_function` does not (it can do anything else):

```
def with_if_statement():
    """
```

```
    >>> with_if_statement()
    1
    """
    if c():
        return t()
    else:
        return f()

def with_if_function():
    return if_function(c(), t(), f())

def c():
    "*** YOUR CODE HERE ***"

def t():
    "*** YOUR CODE HERE ***"

def f():
    "*** YOUR CODE HERE ***"
```

Note: No tests will be run on your solution to this problem.

Question 4

Douglas Hofstadter's Pulitzer-prize-winning book, Gödel, Escher, Bach, poses the following mathematical puzzle.

1. Pick a positive integer n as the start.
2. If n is even, divide it by 2.
3. If n is odd, multiply it by 3 and add 1.
4. Continue this process until n is 1.

The number n will travel up and down but eventually end at 1 (at least for all numbers that have ever been tried — nobody has ever proved that the sequence will terminate). Analogously, a hailstone travels up and down in the atmosphere before eventually landing on earth.

The sequence of values of n is often called a Hailstone sequence, because hailstones also travel up and down in the atmosphere before falling to earth. Write a function that takes a single argument with

formal parameter name n, prints out the hailstone sequence starting at n, and returns the number of steps in the sequence:

```
def hailstone(n):
    """Print the hailstone sequence starting at n and return its
    length.

    >>> a = hailstone(10)
    10
    5
    16
    8
    4
    2
    1
    >>> a
    7
    """
    "*** YOUR CODE HERE ***"
```

Hailstone sequences can get quite long! Try 27. What's the longest you can find?

Question 5: Challenge Problem (optional)

Write a one-line program that prints itself, using only the following features of the Python language:

- Number literals
- Assignment statements
- String literals that can be expressed using single or double quotes
- The arithmetic operators +, -, *, and /
- The built-in print function
- The built-in eval function, which evaluates a string as a Python expression
- The built-in repr function, which returns an expression that evaluates to its argument

You can concatenate two strings by adding them together with + and repeat a string by multipying it by an integer. Semicolons can be used to separate multiple statements on the same line. E.g.,

```
>>> c='c';print('a');print('b' + c * 2)
a
bcc
```

Hint: Explore the relationship between single quotes, double quotes, and the repr function applied to strings.

Place your solution in the multi-line string named challenge_question_program in hw01.py.

Note: No tests will be run on your solution to this problem.