Homework Questions

Q1: A Plus Abs B

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)
```

Use Ok to test your code:

```
python3 ok -q a_plus_abs_b --local
```

Q2: Two of Three

Write a function that takes three positive numbers and returns the sum of the squares of the two largest numbers. Use only a single line 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
```

```
    """
    return _____
```

Use Ok to test your code:

```
python3 ok -q two_of_three --local
```

Q3: Largest Factor

Write a function that takes an integer n that is greater than 1 and returns the largest integer that is smaller than n and evenly divides n.

```
def largest_factor(n):
    """Return the largest factor of n that is smaller than n.

    >>> largest_factor(15) # factors are 1, 3, 5
    5
    >>> largest_factor(80) # factors are 1, 2, 4, 5, 8, 10, 16, 20, 40
    40
    >>> largest_factor(13) # factor is 1 since 13 is prime
    1
    """
    "*** YOUR CODE HERE ***"
```

Hint: To check if b evenly divides a, you can use the expression a % b == 0, which can be read as, "the remainder of dividing a by b is 0."

Use Ok to test your code:

```
python3 ok -q largest_factor --local
```

Q4: If Function vs Statement

Let's 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 ***"
```

To test your solution, open an interactive interpreter

```
python3 -i hw01.py
```

and try calling with_if_function and with_if_statement to check that one returns 1 and the other doesn't.

Hint: If you are having a hard time identifying how the if statement and if function differ, first try to get them to print out different values.

Q5: Hailstone

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

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

This sequence of values of n is often called a Hailstone sequence, 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?

Use Ok to test your code:

```
python3 ok -q hailstone --local
```