Homework 6: Functions

CS-UY 1114 NYU Tandon

Submit your solution to the first problem in a plain text file named according to the pattern *YourNetID*`_hw6_q1.txt`. Submit your solution to *all* remaining problems in a single Python file named *YourNetID*`_hw6.py`.

You should use IDLE to create the Python file. For creating plain text files, you can use Notepad (on Windows), TextEdit (on a Mac), or IDLE. If you are using TextEdit on a Mac, you need to select "Make Plain Text" from the Format menu first.

Problem 1
Before starting the coding assignment, please practice by mentally executing the three following programs and figuring out what they do. You should practice the technique of predicting what Python code does without using Python.

Give the output of each program. That is, your answer should consist only of what each program will `print`.

Do not type the code into your computer!

```
# program1
def x(y):
    print(y * 2)
    return y + 1

def q(v, z):
    z = x(z)
    w = z + x(v)
    print(w)

q(30, 5)

# --------------

# program2
def x(s):
    print(len(s))
    return s[0]+"!"

s = ""
for c in "foo":
    s += x(c)
print(s)
```

```
# --------------

# program3
def a(s):
    return s.upper() * 3

def b(s):
    return chr(ord(s)+1)

def c(s):
    return s[-1]

def d(s):
    return s.lower() * 3

def e(s):
    print("!" + s + "!")

print(a(b(c(d("Foo")))))
```

Problem 2
Consider the following Python function definitions:

```
def print_top (offset) :
    # signature:  int -> NoneType
    print (offset * ' ' + '^')

def print_middle (offset , width) :
    # signature:  int , int -> NoneType
    print (offset * ' ' + '/' + width * ' ' + '\\')

def print_bottom (offset , width):
    # signature:  int , int -> NoneType
    print (offset * ' ' + width * '-')
```

Type these function definitions into your program file.

Experiment with them: write a short program to call these three functions with various parameters until you understand what they do. What is the significance of their parameters?

Write a function called `print_triangle` that takes no arguments, and when called will display the following image.

```
   ^
  / \
 /   \
/     \
-------
```

Your `print_triangle` function must *not* call Python's `print` function directly. It will, however, call the functions `print_top`, `print_middle`, and `print_bottom` listed above in any way you like.

You do not need to use loops in this function.

Test your `print_triangle` function to make sure that it works correctly.

Problem 3
A *rotation* of a string is formed by moving the first character to the end. Thus, the rotation of `"spatula"` is `"patulas"`. The nth rotation of a string is formed by applying a rotation n times. Thus, the 2nd rotation of `"spatula"` is `"atulasp"`.

1. Write a function with this header:

   ```
   def rotate(s):
       """
       sig: str -> str
       Return the rotation of the given string
       rotate("spatula") == "patulas"
       """
   ```

2. Write a function with this header:

   ```
   def rotates(s, n):
       """
       sig: str, int -> str
       Return the nth rotation of the given string
       rotates("spatula", 0) == "spatula"
       rotates("spatula", 1) == "patulas"
       rotates("spatula", 2) == "atulasp"
       """
   ```

   Your solution for `rotates` must be implemented in terms of `rotate`; that is, `rotates` must call `rotate`.

3. Write a function with this header:

   ```
   def all_rotations(s):
       """
       sig: str -> NoneType
       Print all rotations of the given string
       all_rotations("far") should print these 3 lines:
           far
           arf
           rfa
       """
   ```

   Your solution for `all_rotations` must be implemented in terms of `rotates`; that is, `all_rotations` must call `rotates`, but `all_rotations` should not call `rotate`.

Test your solutions thoroughly before submitting them.

Problem 4
Copy the code for the following two functions into your script file:

```
def double (n) :
    # signature:  int -> int
    # return doubled value of parameter
    return n * 2

def succ (n) :
```

```
    # signature:  int -> int
    # returns successor of parameter
    return n + 1
```

Without using any of Python's arithmetic operators (`+, -, *,` etc) complete the bodies of the following functions according to their specifications, by replacing the `pass` statement with appropriate code. You must use the `double` and `succ` functions provided.

```
def f (n) :
    # signature:  int -> int
    # returns the value 2*n + 1
    pass  #  TODO: write the body of this function

def g (n) :
    # signature:  int -> int
    # returns the value 4*n
    pass  #  TODO: write the body of this function

def h (n) :
    # signature:  int -> int
    # returns the value 8*n + 4
    pass  #  TODO: write the body of this function
```

Hint: for completing the `h` function above, you may call the `f` and `g` functions you've already defined.

Call your functions with various inputs. Test your code thoroughly before submitting.

Problem 5
Complete a function named `sum_range` with the following header:

```
def sum_range(start_num, end_num):
    """
    signature: int, int -> int
    precondition: end_num >= start_num
    Returns the sum of all numbers between start_num
    and end_num
    """
```

That is, the function is named `sum_range`; it takes two integer parameters, `start_num` and `end_num`, such that `end_num` is not less than `start_num`; and it returns an integer equal to the sum of all numbers between `start_num` and `end_num`, inclusive.

For example:

```
print(sum_range(5, 5))       # 5
print(sum_range(-4, 4))      # 0
print(sum_range(0, 10))      # 55
print(sum_range(1, 11))      # 66
```

Test your function before submitting it.

Problem 6
Write a function that will count the number of doubled characters in a given string. For the purposes of this problem, a "doubled character" means that the same character is

present at two adjacent positions in the string. If the same character appears three times in a row, that counts as two doubled characters. See the examples below.

Your function should have the following header:

```
def count_doubled(s):
    """
    signature: str -> int
    Returns the number of times that s
    contains a doubled character
    """
```

For example:

```
print(count_doubled("fof"))    # 0
print(count_doubled("foof"))   # 1
print(count_doubled("fooff"))  # 2
print(count_doubled("foooff")) # 3
```

Hint: be careful of off-by-one errors. Make sure that your code doesn't accidentally access an invalid string index.