Rafeh Qazi
Homework 3

Due Friday, October 7 by 4:00pm

As with all homework problems, be careful to show your work and be as specific and clear as possible. For grading convenience, do all of your work on separate pieces of paper, staple them together, and label each page with your name and 'MCS 260: Homework 3' at the top.

1) Consider the following code:

```
a = 2
b = 3

def my_func(x,y):
    r = (x+y)*a
    return r % b

v = my_func(b, a/2)
```

Which variables are local? Which variables are global? What is the "type" of v? Is `my_func` a "pure" function? Fill in the table below, on your own piece of paper, keeping track of the variable values over time. Feel free to leave a box blank when a variable has no value.

	a	b	x	y	r	v
before the call to `my_func(b, a/`	2	3	–	–	–	–
`2)` during the call, before `return`	2	3	3	1	4	–
`r % b` after the call to `my_func(b, a/2)`	2	3	–	–	–	1

2) Recall that list comprehensions are used to quickly create lists that follow some pattern or rule.

a) Consider the mathematical set

$$S = \{\text{integers } x : 1 \leq x < 510510, x \text{ is a multiple of 2 and 510510 is a multiple of } x\}$$

Write a one-line list comprehension in Python syntax which generates this list (you don't have to write down the actual list, just the list comprehension).

>>> [x for x in range(1, 510510) if not x % 2 and not 510510 % x]

b) Using either words or mathematical set notation, describe what numbers the following list comprehension generates. `[x**i for x in list_A for i in range(x)]`

```
let Ai = first element of list_A and A(i+1) be the second element and so on
ex: list_A = [3, 1, 2]
[Ai^0, Ai^1, Ai^2, A(i+1)^0, A(i+2)^0, A(i+2)^1]
```

Note that `list_A` is not defined and you shouldn't define it in your answer (but feel free to test this code using various definitions for `list_A` to see what happens).

c) Define in one line, using a list comprehension, a list called `list_C` that is defined in terms of two other lists `list_A` and `list_B`. For every `a` in `list_A` and every `b` in `list_B`, the new list `list_C` contains the value of `a+b`, but only if `a` and `b` are not equal. Note that `list_A` and `list_B` are not defined and you shouldn't define them in your answer (but feel free to test your code using various definitions for `list_A` and `list_B`).

>>> list_C = [x + y for x in list_A for y in list_B if x != y]

3) Recall that Lambda functions are used to create functions quickly, as well as create "anonymous" functions you don't want to give a name to.

a) In one line, define a function with name `eq_eval` that takes inputs `a`, `b`, `c`, and `x`, and which returns the value of the expression $ax^2 + bx\sqrt{c}\,x{-}1$

```
>>> eq_eval = lambda a, b, c, x: ((a * x) ** 2 + b * x) / ((c*(x)) ** 0.5 - 1)
```

b) Suppose we want to be able to create, on demand, a function of the form f(x) = ax $_2$ + bx + c where the values of `a`, `b`, and `c` will be given later. Define a (non-lambda) function named `poly maker(a,b,c)` which returns a lambda function with only one input `x` and with output

ax $_2$ + bx + c.

```
def polymaker(a, b, c):
        return lambda x: a * x ** 2 + b * x + c
```

c) Consider the following code: `def apply_n_times(input_func, x, n): for i in range(n): x = input_func(x) print(x) apply_n_times(lambda x: 1 - 2 * x, 5, 4)` What lambda should replace the question marks so that the result of the expression $1 - 2 * (1 - 2 * (1 - 2 * (1 - 2 * 5)))$ is printed in the shell? (Hint: what equation is

being applied 4 times to the number 5 in the expression above?)