Homework 4: Recursion and lists

This homework will be done using the hw4.py file available on the class website. The file contains a series of unwritten function definitions. Below are descriptions of what each of the functions should do. You should complete the function definitions so they do the expected thing. Remember to change the `return` statement in the function.

You should test your functions! Some example input-output combinations are given, but you should test on more than just the given examples.

It is always ok to use functions defined in one exercise when writing the functions of later exercises. It is also ok to write your own functions to use in writing these functions. That includes copying over functions from previous homeworks.

When you are done defining these functions, upload the modified hw4.py file in a homework4 folder inside your submission folder. Also print the file and bring it to class.

Exercise 1: `is_power`

This function takes two inputs, both positive integers. It outputs `True` if the first input is an (non-negative) integer power of the second, and `False` otherwise. You must do this exercise using recursion. Do not use any built-in functions other than simple arithmetic. In particular, use nothing in the `math` package.
`is_power(8, 2)` returns `True`.
`is_power(81, 3)` returns `True`.
`is_power(64, 3)` returns `False`.
`is_power(1, 9)` returns `True`.
`is_power(3, 9)` returns `False`.

Exercise 2: `list_reverse`

This function takes as input a list. The output should be a list consisting of the same elements as in the input list, but in reversed order. You must do this exercise using recursion.
`list_reverse([2])` returns `[2]`.
`list_reverse([2, 1, 7, 1, 8])` returns `[8, 1, 7, 1, 2]`.

Exercise 3: `list_primes`

This function takes as input a non-negative integer n and returns a list of the first n primes (in order).
`list_primes(6)` returns `[2, 3, 5, 7, 11, 13]`.
`list_primes(0)` returns `[]`.

Exercise 4: `filter_list`

This function takes as input a list and a test function. (Remember, a "test" is a function that outputs True or False.) It returns a new, possibly shorter list that contains everything in the first list that passes the test (in the same order). In the following examples, `isEven` returns true only for even numbers.
`filter_list(isEven, [1, 2, 3, 5, 4, 2])` returns `[2, 4, 2]`.
`filter_list(isEven, [1, 3, 5])` returns `[]`.

Exercise 5: `list_overlap`

This function takes as input two lists and returns a list consisting of all items that appear in both the input lists. If an item appears more than once in both lists, it should appear more than once in the returned list. More specifically, the number of times an item appears in the output list should be equal to the minimum of the number of times it appears in each of the two input lists. The order of items in the output list is unimportant.
`filter_list([1,2,3,4], [2,4,6,8])` returns `[2,4]`.
`filter_list([1,2,2,4], [2,4,6,8])` returns `[2,4]`.
`filter_list([1,2,2,4], [2,2,4,8])` returns `[2,2,4]`.
`filter_list([1,2,3,4], [5,6,7,8])` returns `[]`.