Question 1.

We will write two functions that take in a list of integers and return True iff the list is sorted. (For example, [5, 7, 7, 10, 23] is sorted and [1, 2, 3] is sorted, but [4, 2, 1] and [5, 4, 6] aren’t sorted.)

Part (a)

Write a function that performs the task using Python’s sorted() function, and then, separately, using list’s sort() method. Be prepared to explain to the TA why one is somewhat more difficult than the other. (Think about list aliasing etc.) Note: to sort a list L in descending order, you can use sorted(L, reverse = True) and L.sort(reverse = True). Do not use while or for loops.

Part (b)

Write a function that performs the task using for or while loops. You may not use Python’s built-in sorting functions.

Question 2.

List elements may be lists themselves. A list with list elements is called a nested list. An example is:

```
pets = ["Shoji", "cat", 18],
    ["Hanako", "dog", 15],
    ["Sir Toby", "cat", 10],
    ["Sachiko", "cat", 7],
    ["Sasha", "dog", 3],
    ["Lopez", "dog", 13]]
```

We can access each element of list pets using its index:

```
>>> pets[3]
["Sachiko", "cat", 7]
```

We can also access elements of the inner lists. For example, since pets[3] refers to a list, we can use pets[3][2] to access the element at position 2:

```
>>> pets[3][2]
7
```

This is saying that element 2 of element 3 of the list pets (the age of the cat named “Sachiko”) is 7.

Part (a)

Write a function that prints each list from the list pets on a separate line.

Part (b)

Write a function that prints the second element (the species) of each inner list in list pets on a separate line.
Part (c)
Write a function that examines the list pets and computes the sum of the ages of the animals in the list. Ages are the third element of the inner lists.

Part (d)
Write a function that examines the list pets and computes the number of dogs in the list.

Question 3.
Write a function called nested_lengths that takes a list $L$ as a parameter and returns a list of the lengths of the sublists of $L$. More formally: for each element $e$ in $L$, the returned list contains a corresponding element, $c$, that represents the number of elements in $e$. Hint:

```python
a = [20, 30]
b = [0, 1, 2, 3]
a.append(len(b))
print(a) # [20, 30, 4]
```

Question 4.

*Challenge question*
A simple (and not very efficient) way to check whether an integer is a prime number is:

```python
def is_prime(n):
    '''Return True iff the integer n is a prime number'''
    import math

    # take care of special cases
    if n < 2:
        return False
    if n == 2:
        return True
    # even numbers larger than 2 are composite
    if n % 2 == 0:
        return False

    # Check every number in 3, 5, 7, etc up to the square root of n
    for i in range(3, int(math.ceil(math.sqrt(n))) + 1, 2):
        if n % i == 0:
            return False
```

# Didn't find any divisors
return True

Part (a)
Write a function with the signature `get_factors(n)` that returns a list of all the prime factors of an integer n. For example, `get_factors(60)` should return `[2, 3, 5]` since $60 = 2^2 \times 3 \times 5$, `get_factors(378)` should return `[2, 3, 7]` since $378 = 2 \times 3^3 \times 7$, and `get_factors(7)` should return `[7]` since 7 is a prime number.

Part (b)
Write a function with the signature `simplify_fraction(frac)` that takes in a list of two integers `frac` which represents the fraction $\frac{frac[0]}{frac[1]}$ and simplifies `frac` “in place”: after `simplify_fraction(frac)` finishes running, $\frac{frac[0]}{frac[1]}$ should retain its value, and `frac[0]` and `frac[1]` should still be integers, and $\frac{frac[0]}{frac[1]}$ should be non-negative and as small as possible. Example:

```python
>> frac = [56, 24]
>> simplify_fraction(frac)
>> print frac #should print [7, 3]
[7, 3]
```

*Hint: the simplest way to accomplish this is by trying to divide by all the possible numbers $\geq 2$ as many times as possible.*