Practice Final Exam

Classes/Objects

1. Define a class called IceCream to help Baskin-Robbins manage their ice cream orders. An IceCream object represents a cone with varying flavors and number of scoops. An IceCream object starts out with no scoop but more can be added.

   The IceCream class should have the following methods.

Method	Description
`__init__(self)`	Creates an IceCream and sets up any fields necessary.
`add_scoop(self, flavor, num_of_scoops)`	Adds the corresponding flavor and number of scoops to IceCream.
`get_flavor(self, flavor)`	Returns the number of scoops of the given flavor in this IceCream. Returns zero if the flavor is not on the cone.
`to_string(self)`	Returns a string representation of the IceCream in the form: “<total number of scoops> scoops of ice cream with <list of flavors>”

2. You are given the following class:

```
class Aquarium:
  def __init__(self, size):
      """
      Creates a new, empty Aquarium of the given size (in gallons)
      """
      self.size = size
      self.fish = {}

  def add_fish(self, new_fish):
      """
      Adds a single fish to the Aquarium
      """
      self.fish[new_fish] = self.fish.get(new_fish, 0) + 1
```

   1. Fill in the following method that will be added inside the Aquarium class.

      ```
      def fish_per_gallon(self):
            """
            Returns the fish per gallon in this
            Aquarium: number of fish divided by number
            of gallons
            """
      ```

   2. Write code that will create an instance of the `Aquarium` class with a size of 60 gallons and assign it to the variable `ruths_aquarium`. This code would appear outside of the class, in a client program.

   3. Write code that will add a 'goldfish' to `ruths_aquarium`. This code would appear outside of the class, in a client program, after the code you have written for Problem 2.2.

Functions

3. Write a function that takes a string and returns the same string but with even indices uppercased and odd indices lowercased. You can use the upper() and lower() function for strings.
   Ex: swap_casing(‘hello world’) should return ‘HeLlO WoRlD’

4. Write a function that takes in a dictionary mapping integers to strings. If the key is divisible by two, set the value equal to “even”. Return a list containing the original values of the even keys.
   Ex: even_key({2:”hello”, 3:”one”, 4: “cat”}) should return [“hello”, “cat”] and the dictionary should look like this {2:”even”, 3:”one”, 4: “even”}

Debugging

5. You receive the following error messages after running your code:

```
Traceback (most recent call last):

File "social_network.py", line 338, in <module>

do_stuff(friends_list)

File "social_network.py", line 200, in do_stuff

result = recommend_by_influence(friendlist)

File "social_network.py", line 107, in
recommend_by_influence

output = read_result()

NameError: global name 'read_result' is not defined
```

   a. List the names of the stack frames that existed at the point that the error was discovered?

   b. What is the most recent stack frame (eg. the last function that was successfully called)?

   c. Describe how you would go about trying to find the cause of and fix the error:

Function Output

6.

Write the output of the code below:

```
sum = 0
for x in range(1, 25, 2):
     temp = (x / 10) % 10
     sum = sum + temp
print('sum:', sum)
```

Documentation

7.

Write a docstring for the following function. Document the inputs, outputs, and any side effects of the function.

```
def unknown_func(num_list):
    '''




    '''
    unique_nums = set(num_list)
    print("There are", len(unique_nums), "unique numbers in the
list")
    factors = {}
    for num in unique_nums:
      factors[num] = set()
      for factor in range(1, num):
            if num % factor == 0:
                factors[num].add(factor)
    return factors
```

List Comprehensions

8.

These two questions refer to this list:

```
lst = [[1, 2, 3, 5], [2, 3, 2], [1, 1, 2, 5, 1], [], [5], [0, 34, 5]]
```

a. Write a list comprehension expression that would produce a list containing just the sublists in lst that contain the value 5. The result should be:

```
[[1, 2, 3, 5], [1, 1, 2, 5, 1], [5], [0, 34, 5]]
```

b. Write a list comprehension expression that would produce a list containing just the maximum value in each sublist in lst. You may use the function max for this question. The result should be:

```
[5, 3, 5, 5, 34]
```

9.

You are given the following class:

```
class Dog:
    def __init__(self, name):
        self.name = name

    def get_name(self):
        return self.name
```

In your main function, you have a list of dog names:

```
dog_names = ['Spot', 'Woofy', 'Fluffy', ""]
```

Write a list comprehension to create a list of Dog objects, one with each name in the list of dog_names, excluding names that are empty strings.