CS 9H Final Exam Review Problems

Python

1 Warm-up

What will Python 2 print?

a) `>>> (2**3, 3) + (1,2)`

b) `>>> "foo" and [] or ""`

c) `>>> "foo" and [] or 42`

d) `for i in "0123":  print i*2`

e) Write a statement that returns `[1, 2, 4, 8, 16, 32]` (list of powers of 2 through 2^5)

i) using `map` and `lambda`.

ii) using list comprehensions.

f)
```
>>> def rearrange(arg): arg = arg[len(arg)//2:] + arg[:len(arg)//2]
>>> a = [2,4,6,8,10]; rearrange(a)
```

2 Data Structures

What does the following function do? What is the structure of the parameter it takes? Give a general explanation, don't just describe each line of code. Provide an example input and output.

```
def wat(foo):
    ret_val = {}
    for thing in foo:
        if thing[0] not in ret_val:
            ret_val[thing[0]] = {}
        for baz in thing[1]:
            if baz not in ret_val[thing[0]]:
                ret_val[thing[0]][baz] = thing[1][baz]
            else:
                ret_val[thing[0]][baz] += thing[1][baz]
    return ret_val
```

3 OOP

a) Design a `Picture` class that holds a 2-dimensional collection of `Pixel` objects. Implement the following methods. The only error handling you need to worry about here is:

- When creating `Pixel` instances, the rgb values must be numbers [0,255]
- when cropping a `Picture`, the new dimensions must be smaller than or equal to the current dimensions (can't crop a `Picture` to be larger).

In these cases, throw an appropriate type of `Exception` with a helpful message.

```
class Pixel:
    def __init__(self, r, g, b):




        self.r =
        self.g =
        self.b =

    def __str__(self):
        return str((self.r, self.g, self.b))

class Picture:


    def __init__(self, pixel_array):
        self.pixels =
        self.length =
        self.width =

    def __str__(self):
        repr = []
        for row in self.pixels:
            for pixel in row:
                repr.append(str(pixel) + "  ")
            repr.append("\n")
        return "".join(repr)
```

```
def crop(self, new_length, new_width):
    """crop the Picture to have the new dimensions
    Ex. if a 3x3 image has pixels at
    0 1 2
    3 4 5
    6 7 9
    calling crop(2,2) would update the image to only contain
    0 1
    3 4"""
```

```
def grayscale(self):
    """convert the Picture to grayscale by setting the RGB values
    of each Pixel to the average of the RGB values of the Pixel.
    Ex. if there is a pixel
    p = Pixel(100, 0, 20)
    in a Picture img, after calling img.grayscale(), the value of p
    would be equivalent to Pixel(40, 40, 40)
    """
```

```
def add_filter(self, filter_name, filter_function):
```

```
    def apply_filter(self, filter):
```

b) You realize that with the endless possibilities of image processing, it would be useful to let programmers to create and use their own filters. Update `Picture` to be able to store and use arbitrary filters. Assume that filters do not take any parameter. Example interaction, assuming that `sharpen` has been defined correctly earlier:

```
>>> img1 = Picture([[Pixel(100,0,20)]*4]*3)
>>> img2 = Picture([[Pixel(20,40, 60)]*5]*5)
>>> img1.add_filter("sharpen", sharpen)
>>> img2.apply_filter("sharpen") # img2 has been sharpened
```

c) What happens if you do

```
>>> pixels = [[Pixel(100,20,60)]*2]*2
>>> img1 = Picture(pixels)
>>> img2 = Picture(pixels)
>>> img1.grayscale()
>>> print img2
```

4 So you think you know Python?

These are just for fun, and are not topics covered on the final. The first is from programmingwats.tumblr.com and the second is via Mehrdad Niknami.

```
>>> def my_append(item, lst = []):
>>>     lst.append(item)
>>>     return lst
>>> print my_append(1)
[1]
>>> print my_append(5, [3, 1, 4, 1])
[3, 1, 4, 1, 5]
>>> print my_append(1)
[1, 1]
```

```
>>> foo = float('nan')
>>> foo in [foo]
True
>>> float('nan') in [foo]
False
>>> foo is foo
True
>>> foo == foo
False
```