Computer Science 134C – Fall 2018 **Anonymous ID:**

Duane A. Bailey

Homework 7 – *Due: November 5*

Please turn in answers to the following class questions on Monday, in class.

1. In lab, we've seen an immutable class, `Color`, that helps us keep track of colors in both `rgb` and `hsv` color spaces. Let's add a couple of methods to this class to provide increased functionality.

   Recall that this is the beginning of the `Color` class definition:

   ```
   class Color(object):
       """An immutable class defining a color using the RGB color model."""

       __slots__ = [ "_red", "_green", "_blue" ]
   ```

   The class has an initializer, as well as `float`-returning property methods `red()`, `green()`, and `blue()`.

   Write an __add(`self,c`)__ method takes another `Color`, `c`, and "adds" it to `self` by mixing, creating a new hybrid `Color`. (Hint: `LighterFilter` could then use a statement like `return c + WHITE`).

   ```
   def __add__(self,c):
   ```

2. Since we designed `Color` objects to be immutable, they can be used as the domain for `dict`s and `set`s. All immutable classes (including `int`, `float`, `str`, and `tuple`) define a __hash__ method that returns a consistent integer value. `Hash` values for equal immutable objects are always the same, while unequal objects have hashes that are *probably* different. This integer value helps organize objects in set-like container classes. How would you go about constructing such a value for `Color`? (Hint: Experiment by calling `hash(o)` on an immutable object, `o`. It's hard to get this wrong!)

   ```
   def __hash__(self):
   ```

3. This week, in lecture, we've seen the Element class. This class carries data in value and a reference, next, to all the Elements that follow it. Write a method of Element, last, that returns the last value in the list reachable from self. (For fullest credit, make last recursive.)

```
def last(self):
    """Return the value of the last element found in the list headed by self."""
```

4. Write a method of Element, sum, that computes the sum (using value's + operator) of all the values encountered from self to the end of the list. (For fullest credit, make sum recursive.)

```
def sum(self):
    """Return the sum of values between self and the end of the list."""
```

⋆