Kerry Ojakian's CSI 32 Class

HW #1

General Instructions:

- Homework must be put in a your dropbox folder; if there are multiple parts, create a single folder for the assignment. Make sure you give clear names to your files and folders.
- Remember that you must work on your own without copying from anyone (that includes classmates and tutors).

The Assignment

1. Write a function `countA` which takes a dictionary as input. You can assume that the values are strings (and the keys may be anything). The function counts the number of values which start with the letter A (lower case or capital). then returns this number.

   Example: If the input to `countA` is the dictionary $\{1 : \text{“}Ace''$, $\text{“}a'' : \text{“}zoo''$, $\text{“}z'' : \text{“}also''\}$, then the function should return 2 because there are two value strings that start with an A: ‘Ace’ and ‘also’.

2. Textbook: Exercise 6.5 (p.233). Also add a setMonth method which takes a single month name as a parameter and resets self._month to the appropriate integer between 1 and 12. If an invalid month is entered, then nothing should be changed.

3. Textbook: Read about the Fraction class in the book then do Exercises 6.10 and 6.11 (p233). Do not use any imported fraction class, just the one in the book. Important: The methods you add should return a Fraction object!

4. Write a class definition **Clock** which models a 24 hour clock; you should include the `init` method and the two methods described below. **Clock** stores a “time” value as internal data, which should always be an integer between 0 and 24 (allowing 0, but not 24); for example, some possible values for time: 12, 0, 23. When a **Clock** is initialized, any non-negative integer can be given as an input to the initializer. **Clock** also has the following methods:

   - `addTime`: Takes one non-negative integer input. This input should be added to the current time stored in the **Clock**, and maintain the time between 0 and 24, adding as we usually do for a clock. For example, if the time is 22 and we called `addTime`(5) the new time is 3 (i.e. 22 + 2 gets us to 0 and then we do +3 more to get to 3).
   - `getTime`: Returns current time.

   Example: The following code should print 22, then 3, then 5.

```
C = Clock(22)
print(C.getTime())
C.addTime(5)
print(C.getTime())
C.addTime(50)
print(C.getTime())
```

   For Extra Credit:

   You will add functionality to **Clock** so that it can keep track of days. It is always initialized to day 0, but every time a 24 hour period is crossed, the number of days increases by one. For example, suppose we are at day 0 and 22 hours; if `addTime`(5) is called, we should be at day 1 and 3 hours. If 50 hours are now added, we are at day 3 and 5 hours. For this extra credit `getTime` should also return days as its first return parameter.

   Example: The same code in the example above following should now print (0,22), then (1,3), then (3,5).