CSCI 1133, Spring 2015 — Assigned: Friday April 10, 2015
Programming Assignment 10 — Due: 11:55pm Thursday April 16, 2015

This is ***not*** a collaborative assignment; you must design, implement and test the solution(s) on your own. You may not consult or work with anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class. Obtaining or *providing* solutions to any homework problems for this class is considered academic misconduct. If you are not sure what this means, consult the class syllabus or discuss it with the course instructor.

This assignment requires writing a single Python module that must be submitted online *prior* to the due date/time. Late submissions will not be accepted. Name your module: `hw10.py` and submit it using the appropriate homework submission link on the Moodle website.

The total point value for programming assignments will be awarded for solutions that are *complete*, *correct*, and *well constructed*. A *"well constructed"* program entails good design, appropriate comments and general readability (descriptive names for variables and procedures, appropriate use of blank space, etc.). The following will result in a score reduction equal to a percentage of the total possible points:

- Incorrectly named/submitted source file (10%)
- Constraints not followed (40%)
- Failure to execute due to syntax errors (30%)

Note that your work will be graded using, and must function correctly with, the current version of Python 3 on CSE Labs UNIX machines. If you complete your programming assignment using a different system, it is your responsibility to ensure your programs work on CSELabs machines *prior* to submitting them.

A. (40 *points*) **Imperial Measurements**

Develop and test a user-defined class named `ImpMeas` that will represent *imperial* distance measurements (i.e., feet and inches). The class will contain a *single* instance variable that represents a linear distance as an integer number of whole inches. The initializer method takes the distance as a string in the following format:

nnn'*mm*"

where *nnn* is the number of whole feet (followed by a single-quote character) and *mm* is the number of inches (followed by a double-quote character). If the distance is omitted, the distance should default to 0'0". Note that either/both of the *feet* or *inches* values may be omitted, and spaces may be embedded anywhere in the string. e.g., the following are all valid distance strings:

```
6'3"
6' 3"
6'
3"
21"
```

Your module should only contain the full class definition. Do not include any test code in your submitted module.

Requirements:

The `ImpMeas` class must include the following:

- `__repr__` and `__str__` methods that return the value of the imperial distance as a string: *feet*'*inches*", where *feet* is the integer value of the number of feet and *inches* is an integer value in [0,11] representing the residual partial feet in inches. e.g., 6'11"

- Provide special methods to support the following arithmetic operations:
    - + : add two imperial measurements and return the distance as an `ImpMeas` object
    - - : return the absolute difference between two imperial measurements as an `ImpMeas` object
    - * : multiply an imperial distance by a *scalar* integer multiplier and return the distance as an `ImpMeas` object
    - / : divide an imperial distance by a *scalar* integer divisor and return the distance as an `ImpMeas` object.
    - % : return the "remainder" (residual feet and inches) of dividing an imperial distance by a *scalar* integer. Return the value as an `ImpMeas` object. For example, if you divide the distance 6' 1" by 3, you obtain three 2' 0" segments with 1" left over (the residual)

Constraints:

- Do not import/use any library modules.

Examples:

```
>>> a = ImpMeas("4'")
>>> b = ImpMeas("2'1\"")
>>> a+b
6'1"
>>> a-b
1'11"
>>> b-a
1'11"
>>> b/2
1'0"
>>> b%2
0'1"
>>> a*4
16'0"
>>> b*4
8'4"
>>> c=ImpMeas("6'")
>>> c
6'0"
>>> c%7
0'2"
>>> c/7
0'10"
>>>
```