CSSE 120 Exam 2

What you should be able to do

This is neither a *contract* nor a *promise*; it is only our *best-effort* to list most of the concepts that you might be expected to demonstrate on this exam.

For the **paper-and-pencil** portion of Exam 2, a student should be able to do problems like the following problems from the ***Practice Problems for the Paper-and Pencil Portion of Exam 2***:

1. Problems **2 and 37**: ***scope***, calling ***functions*** and ***methods*** (including the ***__init__*** method that runs when an object is ***constructed***), accessing ***instance variables***. [Note: there WILL be a problem on the exam like problem 2 and it will count a LOT.]
2. Problem **4**: ***iterating*** though a ***sequence***, building up a sequence with the **+ operator**.
3. Problem **5**: ***function calls***, including functions that call other functions.
4. Problem **6**: ***range*** expressions in all their forms.
5. Problems **18 through 21**: writing short functions "from scratch", especially functions that loop through a list to compute something, find something, or build up a list of somethings.
6. Problems **23 through 31**: ***references*** to objects, the effects thereof, ***mutation***.
7. Problem **38 and 40**: sending ***mutable*** objects (including lists) to functions, using ***box-and-pointer*** diagrams to understand the effects of ***mutation***.
8. Any of the problems from ***Exam 1*** or the practice problems for Exam 1.

For the **on-the-computer** portion of Exam 2, a student should be able to do problems (i.e., test and implement classes, methods and functions) like the following problems from ***Session16_Exam1Practice***:

1. Problem **3a and 3e**: ***iterating through all or part of a sequence***, computing something that is returned (e.g., the sum of parts of some of the items in the sequence).
2. Problem **3b and 3d**: ***finding something in a sequence***, or indicating that it is ***not in the sequence***, and returning the ***found item*** or its ***index*** or other relevant results.
3. Problem **2a, 2b and 3c**: iterating through a sequence or range to ***build up a new sequence, using the + operator***.

   Note: The problems of **Session 12** are excellent examples of additional problems like the above.
4. ***Implement and test a class***, given specifications of the methods. In particular (where all the following examples are taken from implementing the **Box** class in ***problem 1*** of ***Session16_Exam2Practice***:

   a. **Implement the *__init__* method.** (Testing this requires understanding what makes the *__init__* method run.)

      For example, write the *__init__* method for a Box constructed from its *contents* and *volume*.

   b. Implement **methods that have arguments and use *self*** in computing their result.

      For example, the ***append_string*** and ***double*** methods of the Box class.

   c. Implement **methods that mutate *self*** and/or other instances of the class.

      For example, the ***shrink*** method which mutates the Box itself, and the ***steal*** method which mutates both the Box itself and another Box passed as an argument.

d. From within a class, **call other methods of the class,** applied to ***self*** and/or other arguments.

   For example, the ***double_then_shrink*** method which calls ***double***, then ***shrink***, and the ***steal*** method which calls ***append_string***.

e. **Determine what instance variables need to be introduced** to implement a method.

   For example, the ***reset*** method (which required introducing instance variables for the *original contents and volume* of the Box), and the ***get_history*** method (which required introducing an instance variable that is *a list that holds the contents* of the Box at certain points in the Box's lifetime).

f. ***Use and/or mutate*** both ***self*** and ***other arguments that are instances of the class***.

   For example, the ***steal*** method which accessed and set the *contents instance variable* of the *other_box* that was passed as an argument to the *steal* method.

g. Return a ***new instance of the class***.

   For example, the ***combined_box*** method that returned a new Box built from the arguments *self* and *other_box*.

5. Any of the problems from ***Exam 1*** or the practice problems for Exam 1.

6. **Test and debug** any such problems, e.g. by:

   a. Identify a test case that failed.

   b. Work that test case by hand to understand what your code should do.

   c. Trace your code, using *print* statements or the debugger to help you do so, until you find the first place where your code does NOT do what it should do.

   d. Fix the error that you found.

   e. Rinse and repeat until the code passes all test cases AND you have confidence that the code is correct.