Data Structures (CS 1520) Lab #3 Name:_____

Objective: Review classes/objects in Python, and learn how to documentation programs by writing APIs, using preconditions and postconditions, and enforcing preconditions by raising exceptions.

As your programs become more complex (larger in size and scope) and longer lasting (i.e., used for years in a production environment instead of just-run-once "toy" programs), you'll need to pay closer attention to program:

- design - to aid in project development, future maintenance and code reuse
- documentation - to aid fellow developers and future maintainers in understanding and using your software components correctly and effectively
- testing - to aid in software reliability and robustness

To start the lab: Download and extract the file lab3.zip from

http://www.cs.uni.edu/~fienup/cs1520f11/labs/index.htm

The lab3.zip file contains:

- a simple Die class (in the simple_die.py module) for a six-sided die, and
- an AdvancedDie class (in the module advanced_die.py module) for a die which can be constructed with any number of sides. The AdvancedDie class inherits from the Die class.

Part A: Write a program that used AdvancedDie objects to:

- create two 10-sided AdvancedDie objects
- roll the dice 1000 times and tally the total on the pair of dice (i.e., values 2 to 20). HINT: Initialize a list containing 21 zeros, but only use the indexes 2 to 20 to tally dice totals.
- determine which was the most frequent total on the pair of dice and what percentage it occurred.

After you have implemented AND fully tested your program, raise your hand and demonstrate it.

Part B: Complete the Preconditions and Postconditions for the Die and AdvancedDie class methods. For each method with a precondition:

- include code to check the precondition and raise an appropriate exception
- update the corresponding unit test in the unitTestAdvancedDie.py module to test the raising of these exceptions.

After you have implemented AND fully tested your program, raise your hand and demonstrate it.

Part C: For testing certain dice games, suppose we want to extend the AdvancedDie class to include a new method `setRoll` which takes as a parameter a roll value that is used to set a die's roll to a specified value. We might invoke this method as:

```
myDie.setRoll(3)   # sets myDie to a current roll of 3
```

a) Implement a new subclass MoreAdvancedDie (in the module more_advanced_die.py) which inherits from the AdvancedDie class, and includes the `setRoll` method. When implementing the MoreAdvancedDie class:

- add additional method(s), if necessary, to the MoreAdvancedDie class that are needed to check the precondition(s) necessary for the `setRoll` method.
- Include document the MoreAdvancedDie class and its method(s) with enforce preconditions and postconditions
- enforce preconditions by raising appropriate exceptions.

b) View the programmer-authored documentation for the MoreAdvancedDie class by typing help(MoreAdvanceDie) at the Idle shell prompt after importing the MoreAdvancedDie class.

c) EXTRA CREDIT: Use unit testing (unittest module) to test the MoreAdvancedDie class.

After you have implemented AND fully tested your MoreAdvancedDie class, raise you hand and demonstrate it.