Remember that for a function, printing and returning a value are different. You should only print when the problem explicitly says to print or to output.

You should make up *at least* three of your own assertEqual tests for each function BEFORE you write any code. The more you prepare to test your code before you start coding, the more likely you are to discover semantic errors. To determine the correctness of your code for grading purposes, the TA will append and run our own tests.

You MUST use the function names indicated. Half credit will be deducted for any change in function name because of the extra effort the TA will require to grade the assignment.

Read Chapters 8,9,10 in the textbook.

Repeat, read Chapters 8,9,10 in the textbook.

Each problem is worth 5 points. Submit via Sakai a single file named lab3.py with comments separating your answer to each problem below.

1. Write a function "countEvenInts" that returns the number of even **integers** in a list. Note: 14.0 is not an integer, it's a float. Note: consider 0 to be an even integer. (The list may contain any of the atomic types we have learned: integers, floats, strings, Booleans.)

2. Write a function "sumTeenInts" that returns the total of only the teen **integers** (i.e., 13, 14, …, 19) in a list. Float values such as 14.0 and 14.6 should NOT be added. (The list may contain any of the atomic types we have learned: integers, floats, strings, Booleans.)

3. Write a function "insertList" that takes two parameters: a list of integers already in increasing sequential order, and an integer, and returns a new list with the integer inserted in the correct place in the list. Your function should first test the parameters. If the list contains anything but integers, or is not sorted in non-decreasing sequential order, or if the parameter to be inserted is not an integer, your function should output a meaningful error/warning message and return None. Note: the list may have negative and/or duplicate values (e.g., [-9,-3,-3,1,5,5,9,11,11]). Note: the value to be inserted may be negative. Note: the value to be inserted may be less than the smallest value in the list or greater than the largest value in the list. Hint: you may use existing list methods; see Section 5.1 in http://docs.python.org/tutorial/datastructures.html

4. Write a function "pointDistance" with four parameters x1, y1, x2, y2 that computes and returns the distance between two points (x1, y1), (x2, y2) in a Cartesian plane (e.g., the x,y plane). See the following web page for more information: http://math.about.com/library/bldistance.htm Note: Input Testing - if any of the parameters are not integers or floats, print an error message and return None. Your function should work whether the parameters are integers or floats. Note: the parameters may be negative.

5. Write a function “assertAlmostEqual” which has three float parameters. assertAlmostEqual returns True if the first two parameters are less than +/- the third parameter of each other. (The third parameter is called the ‘tolerance’.) Otherwise assertAlmostEqual should return False. Note: if the difference equals the tolerance, return False. Note: Input Testing – all three parameters must be floats; otherwise print an error message and return None.

6. Read about available string methods (see the three .jpg files attached to the assignment). Write a function “wordSeparator” that takes in a sentence where all of the words are run together, but the first character of each word is in upper case. The function should return a converted sentence in which the words are separated by a blank and only the first word starts with an uppercase letter. For example, the sentence “FourScoreAndSevenYearsAgo” would be converted to “Four score and seven years ago” Assume the original sentence has only letters - no punctuation, no white space, no digits. No input testing is required.

7. Write a function “myStringIndex” that takes in 2 arguments: a string and a character, and returns (only) the **first** index whose value is the character. Return None if the character is not in the string. Note: no input testing is required for this function. (For your info, the existing Python method for doing this is stringname.index(char). Do NOT use or even look at this code.)

8. Write a function “addDigits” that takes in a string containing only digits 0-9, and returns an integer that is the sum of the digits. The function should return None for an invalid input. For example: assertEqual (addDigits(“1234”), 10) would succeed

9. (EXTRA CREDIT) Then write a function “isPasswordValid” that takes in a string representing someone’s computer password. To be valid, a password must: (1) be at least 7 characters long, and no longer than 15 characters, (2) contain at least one uppercase letter, (3) contain at least one lowercase letter, (4) contain at least one numeric digit 0 (i.e., 0 thru 9), and (5) contain at least 1 special character in the set {! @ # $ % ^ & *}. Your function should return True or False, and print an appropriate error message when the passwd is invalid.