CS 2316 Individual Homework 2 – Conditionals & Loops
Due: Wednesday, September 4th, before 11:55pm
Out of 100 points

File to submit: HW2.py

Students may only collaborate with fellow students currently taking CS 2316, the TA's, and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc.

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use Piazza

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- ***Do not wait until the last minute*** *to do this assignment in case you run into problems*
- **Read the entire specifications document before starting this assignment.**

Simple Functions

You will write a few python functions for practice with the language. In your HW2.py file, include a comment at the top with your name, section, GTId/Email, and your collaboration statement. Also, include each of the following functions below. For purpose of this homework, you may assume that all inputs will be valid.

1. **lettersToNumbers**
2. **gradeReplacement**
3. **discountShopping**
4. **numDiamond**
5. **horseRace**
6. **suitcasePacker**
7. **nextRow**
8. **turtleBattery**

1. lettersToNumbers(10pts)

Description:
Write a function that takes in one parameter: a string. This string that is passed into your function will have certain letters "converted" to their number representation. The letters that will change and their conversion are as follows: "l" (lower case L) and "I" (upper case i) → "1", "E" → "3", "s" and "S" → "5", "G" → "6", "T" → "7", "g" → "9", "o" and "O" → "0", and "R" → "12". Note with caution which uppercase letters and which lowercase numbers should be converted! The case is very important for this function and uppercase and lowercase letters are different! Your function should find all the occurrences of convertible letters and convert them accordingly. In other words, every time that the convertible letter occurs, replace that letter with its number counterpart. Note that the number counterparts are string representations of said letters.

Parameters:
-aString (String): A string

Return Value:
(String) The new string with all the correct letters replaced

Test Cases:
1. lettersToNumbers("so it goes") returns "50 it 90e5"
2. lettersToNumbers("I like coding") returns "1 1ike c0din9"
3. lettersToNumbers("CS 2316 IS GREAT") returns "C5 2316 15 6123A7"

2. gradeReplacement (10pts)

Description:

Write a function that takes in a list of exam grades and returns the average after performing a grade replacement policy. The grade replacement policy takes the lowest grade from the list and replaces it with the second lowest grade in the list. There will be at least 2 numbers in the list, but there is no upper bound for the length of the list.

Parameters:

-gradeList (List): A list of exam grades as integers

Return:

(Float) The average of all the exam grades after replacement

Test Cases:

1. gradeReplacement([100,90,80,70]) returns 87.5
2. gradeReplacement([100,100,100,100]) returns 100.0
3. gradeReplacement([90, 80, 80, 90, 85]) returns 85.0
4. gradeReplacement([30, 80, 44, 90, 85]) returns 68.6

3. discountShopping (10pts)

Description:
Write a function that takes in a list of raw prices for products that you are buying and returns the total amount of money needed to buy all the items. After having bought 2 items, there is a 5% off discount on the remaining items. After having bought 4 items, there is a 10% off discount on the remaining items. After having bought 8 items, there is a 20% off discount off the remaining items. Note that the discounts do not stack. For example when there are 7 products, the first 2 items are full price, the next 2 items are 5% off, and the final 3 items are 10% off.
Return your floating point answer with no more than 2 digits after the decimal point (it is a dollars & cents answer).

Parameters:
productList (List): A list of product prices

Return Value:
(Float): The total amount needed to be paid

Test Cases:
1. discountShopping([35,20]) returns 55.0
2. discountShopping([10,10,10]) returns 29.5
3. discountShopping([65,28,27,83,67]) returns 257.8
4. discountShopping([1,2,3,4,5,6,7,8]) returns 33.05
5. discountShopping([13,25,42,35,88,76,7,48, 100]) returns 388.25

4. numDiamond (10pts)

Description:
Write a function that takes in the widest row of the diamond as a parameter. The function will then draw a symmetric diamond on screen using the print function. See screenshots below in the test cases for clarification. DO NOT HARD CODE THE PRINTOUTS.

Parameter:
X (Integer): An integer that specifies the widest row of the diamond. You may assume the number is an integer between 2-9.

Return Values:
None

Test Cases:
You have X number of rows, but note that there are three 2s, five 3s, seven 4s, nine 5s, etc.

```
>>> numDiamond(2)
 1
222
 1
>>> numDiamond(5)
    1
   222
  33333
 4444444
555555555
 4444444
  33333
   222
    1
>>> numDiamond(9)
        1
       222
      33333
     4444444
    555555555
   66666666666
  7777777777777
 888888888888888
99999999999999999
 888888888888888
  7777777777777
   66666666666
    555555555
     4444444
      33333
       222
        1
```

5. horseRace(15pts)

Description:
Write a function that will take in a list of length 4 and a float. The list's indexes will each represent a horse with index 0 representing the first horse, index 1 representing the second horse, etc. The number each index can take will be an integer from 1 to 4 inclusive. This integer will represent the user's prediction of a horse's finishing place in a simulated race. For example the list [4,3,2,1] would represent the event that the first horse came in 4th place, the second horse in 3rd place, etc. Note that there can be no repeated integers, and each integer must be used once. The float parameter will represent the amount the user bets that their prediction comes true. The function will simulate a horse race by randomly ordering the list [1,2,3,4]. If the user guesses all four horse's placements correctly then they win double their original bet. (The user can not get 3 correct guesses, because if they do, the 4th one is automatically correct as well!) If the user guesses 2 out of 4 correctly, they win their money back, plus a quarter of the original bet. If the user guesses 1 out of the 4 correctly, they win their money back. If the user guesses none of the places correctly they win no money. The function will return a string stating how many of the horse's placements the user guessed correctly and the award amount from the bet. **HINT**: Use the random module's shuffle method to randomly order lists!

Parameters:
aList (List): A list representing the user's prediction of the place each horse will come in where each index represents a respective horse's place.
aFloat (float): A number representing the users bet (in dollars).

Return Value:
The string “You guessed X position(s) correctly and won $Y!”
OR
The string “You incorrectly guessed every horse's position and lost all of your money.”

Test Cases:
1. 0 correct guesses returns “You incorrectly guessed every horse's position and

lost all of your money!”

2. 1 correct guess with a $100.00 bet returns “You guessed 1 position(s) correctly and won $100.0!”
3. 2 correct guesses with a $45.37 bet returns “You guessed 2 position(s) correctly and won $56.71!”
5. 4 correct guesses with a $5.55 bet returns “You guessed 4 position(s) correctly and won $11.10!”

6. **suitcasePacker (20pts)**

Description:

Write a function that takes in a list of volumes of individual items and a list representing the size (in volume) of two compartments of a suitcase. The method in which you pack this suitcase is as follows: the biggest items are packed first into the bigger compartment followed by the next biggest until you can't fit the next item, then the smallest of the remaining remaining items are packed into the smaller compartment followed by the next smallest, until you can't fit any more items. For example, if you have six items of volumes 1, 3, 9, 4, 2, and 1 respectively and you have a suitcase with compartment volumes of 4 and 15, you will first pack the 9 unit volume item into the 15 unit volume compartment. After, said compartment has 6 units of volume left, so then you pack the 4 unit volume item. Since the 3 unit volume item can't fit into this compartment, we now consider the other compartment and pack the smallest item first, the 1 unit volume item. This is followed by the 1 unit volume item, then the next 1 unit volume item, followed by the 2 unit volume item. Because the 3 unit volume item does not fit in either compartment, it does not get packed. Once your suitcase is packed, generate and return a string: “You have packed X item(s) for a total volume of Y units . The volume of empty space in your suitcase is Z units.” Where X is the number of items you packed, Y is the volume of the items you packed in arbitrary units, and Z is the unused volume in your suitcase. If you have packed every item in your suitcase, the string should be “You have packed every item in your suitcase!” If you can't pack any items in your suitcase then the string should be “You could not pack any items in your suitcase!”

Parameters:

itemList (List): A list of volumes of individual items

suitcaseList (List): A list of volumes of the compartments in your suitcase. This list will always have a length of 2.

Return Value:

A string that describes given condition properly.

Test Cases:

1. suitcasePacker([7,3,1],[7,1]) returns: 'You have packed 2 item(s) for a total volume of 8 units. The volume of empty space in your suitcase is 0 units'
2. suitcasePacker([4,8,3,10,5,5],[19,9]) returns: 'You have packed 4 item(s) for a total volume of 25 units. The volume of empty space in your suitcase is 3 units.'
3. suitcasePacker([13,33,9,1,1,4,19,21,24,13],[60,40]) returns: “You have packed 7 item(s) for a total volume of 85 units. The volume of empty space in your suitcase is 15 units.”
4. suitcasePacker([1,2,3,4,5],[6,10]) returns: 'You packed every item in your suitcase!'
5. suitcasePacker([14,19,9,20],[15,8]) returns: 'You could not fit any items in your suitcase!'

7. nextRow (10pts)

Description:
This is the beginning of pascals triangle:

```
       1
     1 1
    1  2 1
  1  3  3  1
1  4  6  4  1
```

You can calculate any row of Pascal's triangle after the first two from the previous row. Create a new row that starts with a 1, filling in the inner values such that each number is the sum of the two values to the upper left and upper right above t in the previous row, and then adding a 1 to the end.

For example, the 3rd row of Pascal's triangle is [1,3,3,1]. So the 4th row would be calculated as [1, 1+3, 3+3, 3+1,1] = [1,4,6,4,1].

Write a function called **nextRow** that takes in one parameter, a list of numbers representing a row in a Pascal's triangle (of at least 2 numbers) and return a list representing the next row.

Parameters:
aRow – A row of integer numbers.

Return Values:
A list of numbers representing the NEXT row of the triangle.

Examples:
nextRow([1,1]) returns [1,2,1]
nextRow([1,2,1]) returns [1,3,3,1]

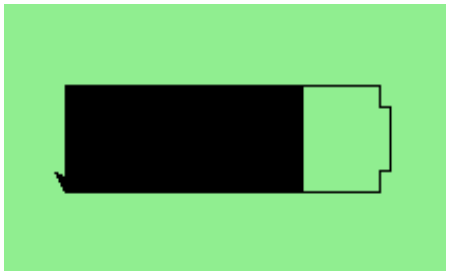
8. turtleBattery (15pts)

Description:
Write a function that uses the turtle module to draw a battery with a given length and percentage of the battery filled/"left". The height of the battery should always be 50. The end "cap" of the battery should always have dimensions 5 length by 30 high. The "cap" of the battery will NOT be filled in, even if the battery is 100% full. See the example pictures below for clarification.

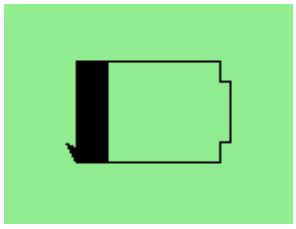
Parameters:
length (Integer): An integer representing the length (in pixels) of your battery body (not including the cap)
percentage (Integer): An integer between 0 and 100 representing the percentage of the battery body your turtle will fill.

Return Values:
None

Examples:
turtleBattery(150,75)



turtleBattery(70,22)



Grading Rubric

lettersToNumbers(10pts)

- Finds all letters in the string that need to be converted. 8pts
- Returns the correct string with the replaced letters. 2pts

gradeReplacement(10pts)

– Correctly replaces the lowest value with the second lowest value 5pts
- Returns correct average 5pts

discountShopping(10pts)

- Correctly applies discounts to appropriate items 5pts
- Returns correct total amount 5pts

numDiamond(10pts)

- Correct number of rows and correct number in rows 5pts
- Correct shape (-5 if hard coded) 5pts

horseRace(15pts)

- Takes in correctly formatted parameters 1pts
– Imports random and simulates the race 4pts
– Returns proper “You incorrectly guessed every horse's position and lost all of your money!” 3pts
– Returns proper “You guessed X positions correctly and won $Y!” 6pts

suitcasePacker(20pts)

- Takes in two correctly formatted parameters 1pt
- Returns “You have packed X item(s) for a total volume of Y units .” for standard case 5pts
– Returns “The volume of empty space in your suitcase is Z units.“ for standard case 4pts
– Returns “You could not pack any items in your suitcase!” I f no items are packed 5pts
– Returns “You have packed every item in your suitcase!” if all items are packed 5pts

nextRow(10pts)

– Returns correct nextRow for all possible input rows 10pts

turtleBattery(15pts)

- Takes in two parameters 1pt
- Apparent battery shape 3pts

- Correct amount of battery filled 5pts
- Battery length coordinates with length parameter 3pts
- Percentage of battery filled coordinates with pct parameter 3pts