CS 2316 Individual Homework 2 – Conditionals & Loops
Due: Tuesday, May 28th, 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. **replaceLetter**
2. **gradeReplacement**
3. **discountShopping**
4. **numMountainRange**
5. **guessingGame**
6. **scheduler**
7.
8. **clockTurtle**

1. replaceLetter(10pts)

Description:
Write a function that takes in three parameters: a string that consists of one letter (the letter that will be replaced), a second string that consists of one letter (the replacement letter), and a string. Your function should find all the occurrences of your first parameter in the string. Every time that the first parameter letter occurs, replace that letter with the second parameter‘s letter. Note that uppercase letters and lowercase letters are considered different letters.

Parameters:
-oldLet (String): The letter you want to replace
-newLet (String): The letter that will replace oldLet
-aStr (String): A string

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

Test Cases:
1. replaceLetter("e", “a“, “I like CS2316!”) returns “I lika CS2316!”
2. replaceLetter("L", “c“, “HELLO world”) returns “HEccO world”
3. replaceLetter(“Q”, “R”, “MY NAME IS MILES”) returns “MY NAME IS MILES”

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 numbers in 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 3 items, there is a 10% off discount on the remaining items. After having bought 5 items, there is a 15% 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 3 items are full price, the next 2 items are 10% off, and the final 2 items are 15% off.
Return your floating point answer with only 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 30.0
3. discountShopping([65,28,27,83,67]) returns 255.0
4. discountShopping([1,2,3,4,5,6,7,8]) returns 31.95
5. discountShopping([13,25,42,35,88,76,7,48, 100]) returns 382.05

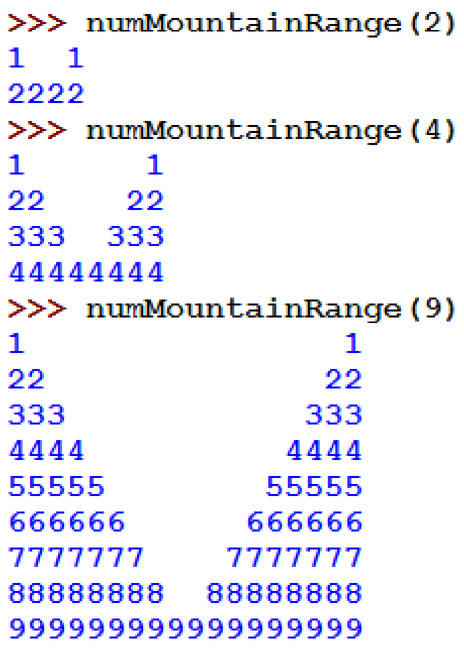
4. numMountainRange (10pts)

Description:
Write a function that takes in the number of rows of the mountain range as a parameter. The function will then draw a number mountain range 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 number of rows of the mountain range. 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 two 1s, four 2s, six 3s, eight 4s, etc.



```
>>> numMountainRange(2)
1  1
2222
>>> numMountainRange(4)
1      1
22    22
333  333
44444444
>>> numMountainRange(9)
1                1
22              22
333            333
4444          4444
55555        55555
666666      666666
7777777    7777777
88888888  88888888
999999999999999999
```

5. **guessingGame(10pts)**

Description:
Write a function that will take in 4 boolean parameters and return a string that tells the user if they won the game and what prize they receive. The function will generate 4 random boolean values and compare then with the users guesses. If none of them match, return the string "You missed all the guesses and lost the game". Otherwise, return the string "You guessed correctly X times and won a Y". Replace X in the above string with the number of correct guesses by the user. Replace Y with the following prizes: 1 guess is a "tablet", 2 guesses is a "Mac Laptop", 3 guesses is a "Linux Laptop", and all 4 guesses is a "Windows Laptop". **HINT**: Use the random module to generate a 0 or 1 and convert these to booleans.

Parameters:
guess1: a boolean (True or False) representing the users first guess
guess2: a boolean (True or False) representing the users second guess
guess3: a boolean (True or False) representing the users third guess
guess4: a boolean (True or False) representing the users fourth guess

Return Value:
The string "You guessed correctly X times and won a Y" OR
The string "You missed all the guesses and lost the game"

Test Cases:
1. 0 correct guesses returns "You missed all the guesses and lost the game"
2. 1 correct guess returns "You guessed correctly 1 times and won a tablet"
3. 2 correct guesses returns "You guessed correctly 2 times and won a Mac Laptop"
4. 3 correct guesses returns "You guessed correctly 3 times and won a Linux Laptop"
5. 4 correct guess returns "You guessed correctly 4 times and won a Windows Laptop"

6. **scheduler (20pts)**

Description:
Write a function that takes in a list of hours for each activity and the number of hours left in the day. You are trying to participate in activities for as long as possible for the rest of the day. To do this, always attend the activities with the largest time first before attending activities that will take less time. For example, if you have 8 hours left and activities lasting 10, 1, 3, and 4 hours, you want to attend activities lasting 4, 3, and 1 hours because there is not enough time in the day to attend the 10 hour activity. If you have 10 hours left with the same activities, you want participate in the 10 hour activity to fill up your day. Generate a string: “You have attended X activities for Y hours. There are Z hours left in the day.“ where X is the number of activities you attended, Y is the amount of hours you spent participating in activities, and Z is the remaining hours you have available in the day. If you have attended every activity for the day, the string should be “You have attended every activity today.” If you can’t attend any activities because there is not enough time left in the day, return the string “You cannot attend any activities.”

Parameters:
aList (List): A list of hours for activities
aNum (Integer): Integer that represents the number of hours left in the day

Return Value:
A string that describes given condition properly.

Test Cases:
1. scheduler([10,1,3,4],8) returns: 'You have attended 3 activities for 8 hours. There are 0 hours left in the day.'
2. scheduler([22,3,7],9) returns: 'You have attended 1 activities for 7 hours. There are 2 hours left in the day.'
3. scheduler([10,1,2,3,8],8) returns: 'You have attended 1 activities for 8 hours. There are 0 hours left in the day.'
4. scheduler([10,9,8,11], 5) returns: 'You cannot attend any activities.'
5. scheduler([1,5,3,4,3,1,5], 25) returns: 'You have attended every activity today.'

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]

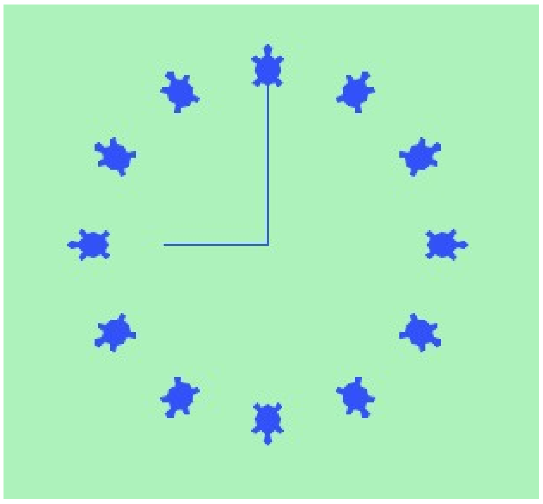
8. clockTurtle (20pts)

Description:
Write a function that uses the turtle module to draw a clock with a given clockHour, as short hand of the clock, and aNum, as the radius. You may assume that the long hand of the clock will stay at 12 at all times. You do need to draw the clock layout using turtle module. At each hour position (12, 1, 2, 3, 4, 5, etc), make your turtle leave a stamp of itself. (You can change the turtle shape if you want). **NOTE**: This is a case where doctest is not able to actually test this function. Therefore, there will be no required tests for this function.

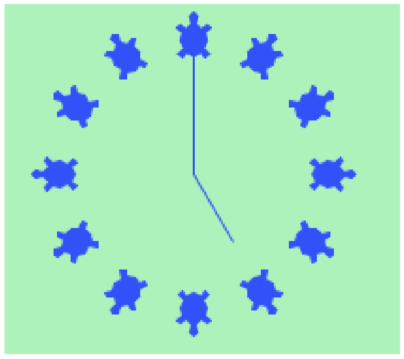
Parameters:
clockHour (Integer): an integer between 1 and 12 representing the short hand of the clock
aNum (Integer): radius of the clock

Return Values:
None

Examples:
clockTurtle(9,100)



clockTurtle(5,70)



Grading Rubric

replaceLetter(10pts)

- Finds all letters in the string that need to be replaced.	5pts
- Returns the correct string with the replaced letters	5pts

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

numMountainRange(10pts)

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

guessingGame(10pts)

- Takes in four parameters	1pts
- Imports random and generates 4 booleans	2pts
– Returns proper “You missed all the guesses and lost the game”	2pts
- Returns proper “You guessed correctly X times and won a Y”	5pts

scheduler(20pts)

- Takes in two parameters	1pt
- Returns “You have attended X activities for Y hours” for standard case	5pts
- Returns “There are Z hours left in the day.“ for standard case	4pts
- Returns 'You cannot attend any activities' if not enough hours	5pts
– Returns 'You have attended every activity today!' if all activities attended	5pts

nextRow(10pts)

– Returns correct nextRow for all possible input rows	10pts

clockTurtle(20pts)

- Takes in two parameters	1pt
- Long hand stays at 12	3pts
- Short hand is drawn at correct hour position	7pts
- Short hand is shorter than Long hand (recognizably)	2pts
- Turtle is tamped at each hour position	2pts
- Turtles are stamped correctly (in correct direction)	3pts
- The clock size changes as radius changes	2pts