CS 2316 Individual Homework 2 – Conditionals & Loops
Due: Wednesday, Sep 5th, 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. **countLetter**
2. **movieAge**
3. **curveGrade**
4. **numPyramid**
5. **appleFan**
6. **groupSeater**
7. **clockTurtle**

1. countLetter(10pts)

Description:

Write a function that takes in a string that is one character and a 2nd string as parameters. Your function should find how many times the letter appears in the string. Note that it should consider both capital and lower case letters as matching!

Parameters:
-aLet (String): A string that is the letter you are trying to find
-aStr (String): A string

Return Value:
(Integer) Number of times the letter appears in the string.

Test Cases:
1. countLetter("e", “I like CS2316!”) returns 1
2. countLetter("L", “Hello world”) returns 3
3. countLetter(“m”, “MY NAME IS MILES”) returns 3

2. movieAge (10pts)

Description:
Write a function that takes in a list of ages and returns the total amount needed to get into the movie theater. If you are 13 or younger, you need to pay $8. If you are older than 13, you need to pay $11. If you are 60 or older, you get a senior citizen discount, and you will need to pay $9. Return a string that has a dollar sign at the front!

Parameters:
aList (List): A list of people’s ages

Return Value:
totalAmount (String): The total amount need to be paid

Test Cases:
1. movieAge([65,12,23,30]) returns “$39”
2. movieAge([10,13,20,60]) returns “$36”

3. curveGrade (15pts)

Description:
Write a function to curve (add) points on each exam grade and return the overall average of the class after the grades are curved. **The maximum points a student can get is 100, even with the curve points.**

Parameters:
aList (List): A list of grades, ranging from 0 to 100
curvePt (Integer): The amount each exam should be curved

Return Value:
(Float) Overall average of all exams in the list after curving. .

Test Cases:

1. curveGrade([89,40,73,90,55,69],5) returns 74.33333333333333
2. curveGrade([30,50,98,79,82],10) returns 76.2
3. curveGrade([100,73,49,95,20,37,83,74],13) returns 76.75

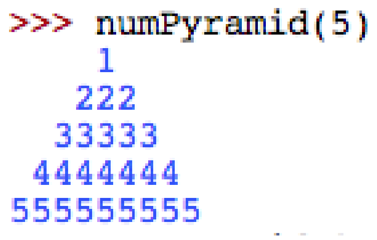
4. numPyramid (10pts)

Description:
Your function will draw a number pyramid on screen using the print function. Your number pyramid will have X rows.

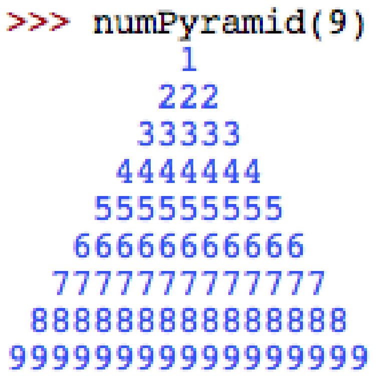
Parameter:
X (Integer): An integer that specifies the number of rows of the pyramid. You may assume the number is an integer between 1-9.

Return Values:
None

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



```
>>> numPyramid(5)
    1
   222
  33333
 4444444
555555555
```



```
>>> numPyramid(9)
        1
       222
      33333
     4444444
    555555555
   66666666666
  7777777777777
 888888888888888
99999999999999999
```

5. **appleFan (15pts)**

Description:
Write a function that returns a string based on the apple products selected by the inputs. Use the inputs True and False. The function should return the string “I have ” concatenated with the designated apple products. The four apple products should be: "IPods", "ITouchs", "IPhones", and "Macs". If the user doesn’t have any Apple products, return the string “Not an Apple fan.” If the user has all four Apple products, return the string “I’m a huge Apple fan!”

Parameters:
answer1: a boolean (True or False) representing whether the user has “IPods”
answer2: a Boolean (True or False) representing whether the user has “ITouches”
answer3: a boolean (True or False) representing whether the user has “IPhones”
answer4: a boolean (True or False) representing whether the user has “Macs”

Return Value:
The string “I have” + the designated Apple products + “.”

Test Cases:

1. appleFan(True, True, True, True) --> “I have IPods, ITouchs, IPhones, Macs. I’m a huge Apple fan!”
2. appleFan (True, False, True, False) --> “I have IPods, IPhones.”
3. appleFan (False, False, False, False) --> “Not an Apple fan.”

6. **groupSeater (20pts)**

Description:
Write a function that takes in a List of parties and number of seats available as parameters. You are trying to seat as many parties as possible with the given groups. For example, if you have 8 seats and parties of 10, 1, 3, and 4, you want to seat parties of 1, 3, and 4 first, instead of having to wait for 2 more seats for the party of 10. If you have 10 seats and the same parties, you want to seat the party of 10 first. Always seat the groups with the most people first (if they fit) before moving on to the smaller groups! Generate a string: “You have seated parties of” and concatenate it with the parties you have seated as well as the number of seats remaining. If you have seated EVERYONE waiting in line, the string should be “You have seated everyone!” If you can’t seat anyone because there are no parties that can get seated with available seats, return “You cannot seat anyone. Wait for more seats to be available!”

Parameters:
aList (List): A list of parties of people
aNum (Integer): Integer that represents number of seats available

Return Value:
A string that describes given condition properly.

Test Cases:
1. groupSeater([10,1,2,3,7],8) returns: “You have seated parties of 7, 1. There are 0 seats left.”
2. groupSeater([10,9,8,11], 5) returns: “You cannot seat anyone. Wait for more seats to be available!”
3. groupSeater([1,5,3,4,3,1,5], 25) returns: “You have seated everyone!”

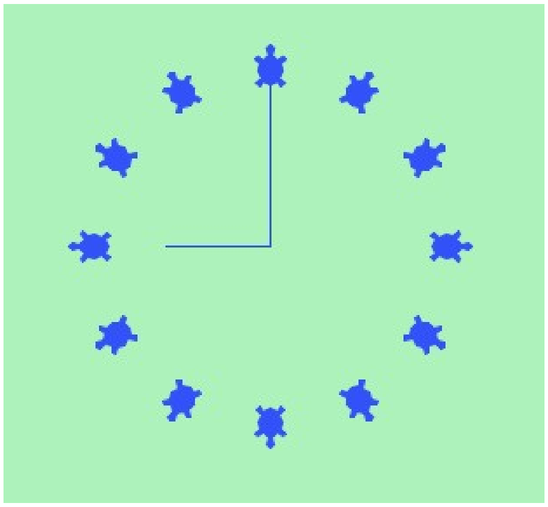
7. **clockTurtle (20pts)**

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.)

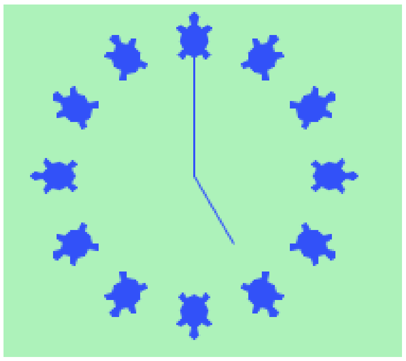
Parameters:
clockHour (Integer): an integer between 1 and 12 representing the short hand of the clock
aNum (Integer): radius of the clock

Return Values:
None

Examples:
clockTurtle(9,100)



clockTurtle(5,70)



Grading Rubric

countLetter(10pts)

- Finds all letters in both lower and upper case letters.	5pts
- Returns correct number	5pts

movieAge(10pts)

- Correctly limits age and converts to correct price	5pts
- Returns correct total amount	5pts

curveGrade(15pts)

- Limits grade above 100 while calculating average	5pts
- Returns correct average	10pts

numPyramid(10pts)

- Correct number of rows and correct number in rows	5pts
- Correct shape	5pts
-	

appleFan(15pts)

- Takes in four parameters	1pt
- Returns proper “I have”+products+”.”	4pts
- Returns “Not an Apple fan if False for all parameters	5pts
- Returns “I’m a huge Apple fan!” in addition to “I have” string	5pts
-	

groupSeater(20pts)

- Takes in two parameters	1pt
- Returns “You have seated parties of” string for excess seats	4pts
- Returns “There are x seats left” with correct number of seat left	5pts
- Returns “You cannot seat anyone….” If not enough seats	5pts
- Returns “You have seated everyone!” if all parties are seated	5pts
-	

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

Homework Created by: Catherine Hwang – Spring 2012