CS 1301 Individual Homework 3 – Conditionals & Loops
Due: Friday, September 13th, before 11:55pm
Out of 100 points

File to submit: HW3.py

Students may only collaborate with fellow students currently taking CS 1301, 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.**

Functions
You will write a few python functions for practice with the language. In your HW3.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.

Function Name: **passOrNot**

Parameters:

grade – a number representing the user's grade as an integer or float

Return Value:

Either the string “Congratulations. You passed!” or the string “Sorry. You must have at least 70% to pass. See you next semester.”

Description:

Write a function for your CS1301 class that determines whether the user passes the class or not. If the user's grade, which is provided by the parameter grade, is greater than or equal to the minimum grade (70), return the string 'Congratulations. You passed!'. Otherwise, return the string “Sorry. You must have at least 70 percent to pass. See you next semester.”

Test Cases:

passOrNot(42.5) --> “Sorry. You must have at least 70% to pass. See you next semester.”
passOrNot(89) --> “Congratulations. You passed!”

Function Name: **modulusFour**

Parameters:

start - an integer greater than 0 representing the number that will be modulated by four

Return Value:

None

Description:

Write a function to show how many times four divides into a number. The function should print the numbers from the given number to the remainder (decreasing by 4 each time...if you go below 0, don't print it!) in descending order, with each number being printed on its own line. After printing the required numbers, on a separate line, print the string “The remainder is the number shown above.”

Test Cases:

```
 >>> modulusFour(5)
5
1
The remainder is the number shown above.

>>> modulusFour(27)
27
23
19
15
11
7
3
The remainder is the number shown above.

>>>modulusFour(2)
2
The remainder is the number shown above.
```

Function Name: **letterSpace**

Parameters:

userString - A String.

Return:

A String.

Description:

Write a function that uses a while loop to create and return a new string that contains only the letters of the original input, leaving a space in the place of numbers, punctuation, and symbols.. If the input string has no letters, you must return a string of spaces.

You **MUST use a while loop** for this problem! Hint: "import string" and use the "in" check along with the "string.ascii_letters" constant to determine if each character is a letter or not.

Test Cases:

```
 >>> x = letterSpace("gburdell3")
 >>> print(x)
gburdell

 >>> y = letterSpace("Hello@World.com")
>>> print(y)
Hello World com

>>> letterSpace("2013")
'    '
```

Function Name: **complimentMaker**

Parameters:

answer1 – a boolean (True or False) representing whether the user is "super"
answer2 - a boolean (True or False) representing whether the user is "nice"
answer3 - a boolean (True or False) representing whether the user is "smart"
answer4 - a boolean (True or False) representing whether the user is "cool"

Return Value:
The string “You are ” + the designated compliments + “.”

Description:
Write a function that **returns** a string of compliments based on the adjectives selected by the inputs. Use the inputs True and False. The function should return the string “You are ” concatenated with the compliments that are true. The four compliments should be: "super" "nice" "smart" and "cool". If none of the compliments are true, return the string “No comment.” instead.

Test Cases:

1. complimentMaker(True, True, True, True) --> “You are super nice smart cool.”
2. complimentMaker(True, False, True, False) --> “You are super smart.”
3. complimentMaker(False, False, False, False) --> “No Comment.”

Function Name: **wordMesh**

Parameters:

wordA – a string
wordB – a string

Return Value:

aString – With the correct value.

Description:
Write a function that takes in a two strings. Have your function **return** out the two words as one meshed word, with the characters alternating between the first word and the second word. Assume the user will input words of equal length.

Test Cases:

```
>>> x = wordMesh("HELLO","world")
>>> print(x)
HwEoLrLlOd

>>> wordMesh("cat","DOG")
'cDaOtG'

>>> wordMesh("GOLD","fish")
'GfOiLsDh'
```

Function Name: replaceWord(10pts)

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

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

Description:
Write a function that takes in three parameters: a string that consists of one word (the word that will be replaced), a second string that consists of one word (the replacement word), 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 word with the second parameter’s word. Note that uppercase letters and lowercase letters are considered different letters. *HINT: Look at the .replace method in the string object!*

Test Cases:

```
>>> replaceWord("Jack", "Jill", "Jack and Jill went up the hill to fetch Jack some water.")
‘Jill and Jill went up the hill to fetch Jill some water.’

>>> replaceWord("hard", "easy", "That CS test was so hard I wanted to cry.")
‘That CS test was so easy I wanted to cry.’

>>> replaceWord("Jingle", "Tinker", "I dropped my jingle bell!")
‘I dropped my jingle bell!’
```

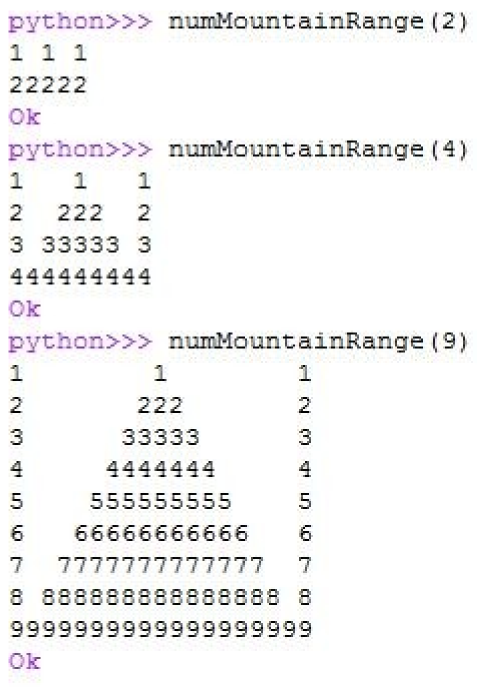
Function Name: numMountainRange (10pts)

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, you should have one set of code that will work for any number

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 three 1s, five 2s, seven 3s, nine 4s, etc.



```
python>>> numMountainRange(2)
1 1 1
22222
Ok
python>>> numMountainRange(4)
1   1   1
2  222  2
3 33333 3
444444444
Ok
python>>> numMountainRange(9)
1        1        1
2       222       2
3      33333      3
4     4444444     4
5    555555555    5
6   66666666666   6
7  7777777777777  7
8 888888888888888 8
9999999999999999999
Ok
```

Function Name: **print10table**

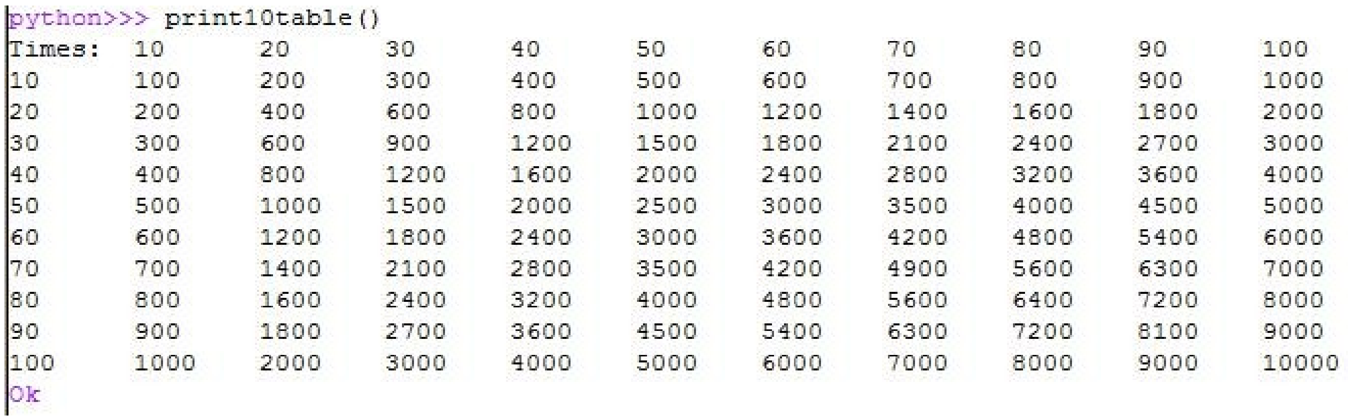
Parameters:

    none

Return Value:

    none

You are hired to develop an educational software package. Your first job: Write a function print10table() that will *print* the times tables (up to 100, by increments of 10) on the screen. When your function is called, it should print the following:



```
python>>> print10table()
Times:  10      20      30      40      50      60      70      80      90      100
10      100     200     300     400     500     600     700     800     900     1000
20      200     400     600     800     1000    1200    1400    1600    1800    2000
30      300     600     900     1200    1500    1800    2100    2400    2700    3000
40      400     800     1200    1600    2000    2400    2800    3200    3600    4000
50      500     1000    1500    2000    2500    3000    3500    4000    4500    5000
60      600     1200    1800    2400    3000    3600    4200    4800    5400    6000
70      700     1400    2100    2800    3500    4200    4900    5600    6300    7000
80      800     1600    2400    3200    4000    4800    5600    6400    7200    8000
90      900     1800    2700    3600    4500    5400    6300    7200    8100    9000
100     1000    2000    3000    4000    5000    6000    7000    8000    9000    10000
Ok
```

Note that your function must print a header (Times: 10...100) and a first column number that goes from 10…100, while the interior of the grid is the X * Y value. Hint: Using two loops (one inside of the other) is an easy (but not the only) way to accomplish this. You may want to use tab characters ("\t") to space your grid out correctly.

Function Name: **printTimes**

Parameters:

N – an integer that limits the upper bound of the times table (inclusive)

inc – a positive integer (either 1 or 2) that decides increment

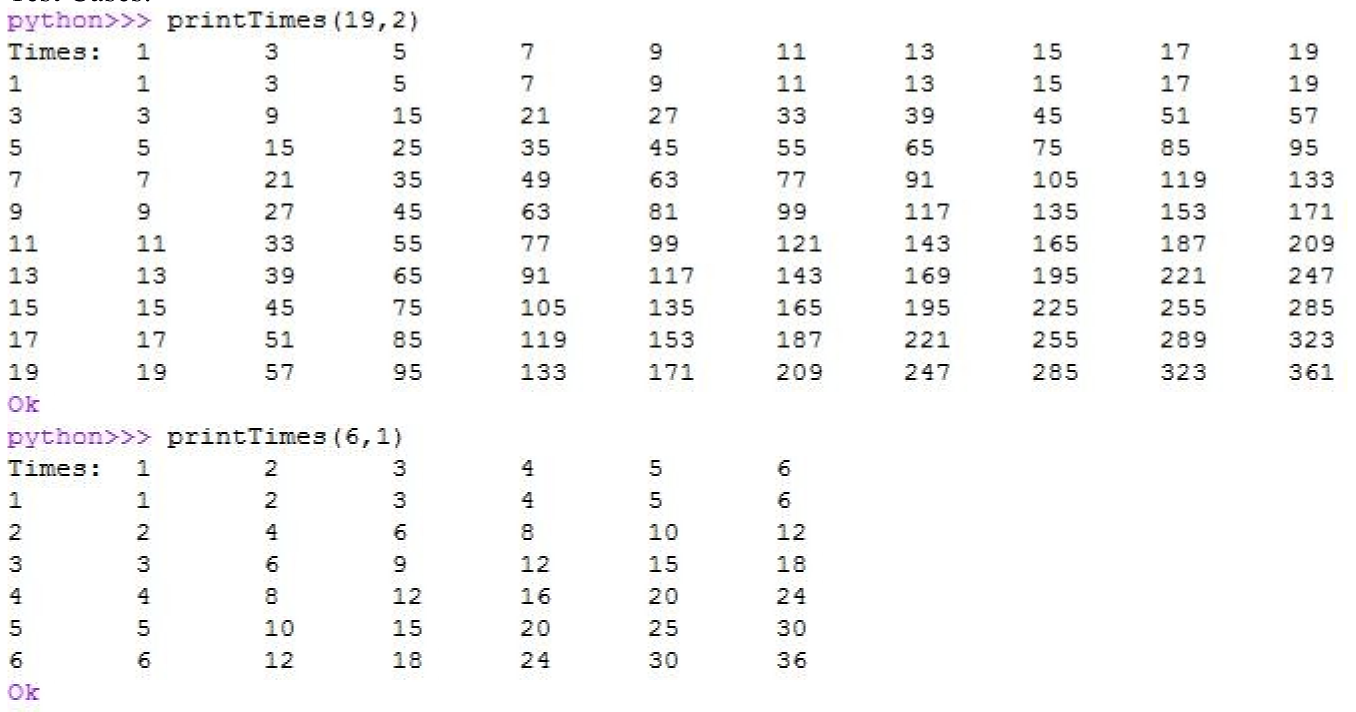
Return Values:

none

Description

Your boss was impressed with your 100x100 times table function. Now he wants you to modify the function so that it will work for any sized times table, increasing by increments of 1 or 2. Write a printTimes(N , inc) function that will print a times table from 1 up to N by increments of inc, for any positive number N. Note: one parameter must be an odd number, and the other an even number.

Test Cases:



```
python>>> printTimes(19,2)
Times:  1       3       5       7       9       11      13      15      17      19
1       1       3       5       7       9       11      13      15      17      19
3       3       9       15      21      27      33      39      45      51      57
5       5       15      25      35      45      55      65      75      85      95
7       7       21      35      49      63      77      91      105     119     133
9       9       27      45      63      81      99      117     135     153     171
11      11      33      55      77      99      121     143     165     187     209
13      13      39      65      91      117     143     169     195     221     247
15      15      45      75      105     135     165     195     225     255     285
17      17      51      85      119     153     187     221     255     289     323
19      19      57      95      133     171     209     247     285     323     361
Ok
python>>> printTimes(6,1)
Times:  1       2       3       4       5       6
1       1       2       3       4       5       6
2       2       4       6       8       10      12
3       3       6       9       12      15      18
4       4       8       12      16      20      24
5       5       10      15      20      25      30
6       6       12      18      24      30      36
Ok
```

Grading Rubric

passOrNot		**5pts**
- function takes in a grade	2	
- function returns correct output for all valid inputs	3	
modulusFour		**5pts**
- function prints numbers starting at specified parameter	2	
- function print decreases by 4 every time	2	
- function stops printing at proper value	1	
letterSpace		**10pts**
- uses a while loop	4	
- returns correct output for any valid input	6	
complimentMaker		**10pts**
- function accepts parameters as booleans	4	
- function correctly generates string output	6	
wordMesh		**10pts**
- function accepts two parameters	5	
- function correctly meshed word	5	
replaceWord		**10pts**
- Finds all letters in the string that need to be replaced.	5	
- Returns the correct string with the replaced letters	5	
numMountainRange		**20pts**
- Correct number of rows and correct number in rows	10	
- Correct shape (-5 if hard coded)	10	
Print10table		**10pts**
- function prints correct multiplication output	5	
- function prints with correct formatting	5	
printTimes		**20pts**
- function accepts an integer *n* and *inc* as paremeters	5	
- function correctly prints *n x n* times table	5	
- function nicely formats the output	5	
- function does not return any value	5	

Elements of this homework created by Catherine Hwang and James Moore and Alec Kaye