CS 1301

Individual Homework 3 – Conditionals & Loops

Due: Friday, February 11th, before 6 PM
Out of 100 points

Files to submit: **1. HW3.py**

THIS IS AN INDIVIDUAL ASSIGNMENT!

You should work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

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.

Part 1 – Simple Functions

You will write a few python functions for practice with the language. In your submission file, include a comment at the top with your names, section, GTID/Email, and your collaboration statement. Also include each of the following functions.

1. checkHeight
2. smallestAndLargest
3. countDownByTwos
4. multiplicationTables
5. complimentMaker
6. comboLock
7. badRecord
8. printTimestable
9. printTimes

Function Name: **checkHeight**

Parameters:

height - an integer representing the user's height in centimeters

Return Value:

Either the string “Have a great ride!” or the string “Sorry. You must be at least 1 meter 32 cm to ride.”

Test Cases:

checkHeight(125) --> “Sorry. You must be at least 1 meter 32 cm to ride.”

checkHeight(169) --> “Have a great ride!”

Description:

Write a function for the superman ride at six flags that determines whether the user is taller than 1 meter 32 centimeters so that he or she can ride a roller coaster. If the user's height, which is provided by the parameter height, is greater than or equal to the minimum height, **return** the string 'Have a great ride!'. Otherwise, **return** the string 'Sorry. You must be at least 1 meter 32 cm to ride.'

Function Name: **smallestAndLargest**

Parameters:

num1 - a floating point number being compared against num2 and num3

num2 - a floating point number being compared against num1 and num3

num3 - a floating point number being compared against num1 and num2

Return Value:

A string formatted exactly as follows: "Smallest Number: {smallest}. Largest Number: {largest}" where {smallest} and {largest} should be the value of num1, num2, or num3.

Test Cases:

```
>>> largestAndSmallest(0,0,0)
Largest Number: 0.000000; Smallest Number: 0.000000
>>> largestAndSmallest(23.4, 2, 23.4)
Largest Number: 23.400000; Smallest Number: 2.000000
>>> a = largestAndSmallest(-4, 30.768, 23)
>>> a
Largest Number: 30.768000; Smallest Number: -4.000000
```

Description:

Write a function that takes in three numbers as parameters and **returns** a string describing the smallest and largest number of the inputted parameters. If two or all of the numbers are equal, or even all three, then just display one of them. The function may not use any built-in math functions, e.g. you may not use max(). Note: If a situation where 3.7 is represented by 3.7000000002 or any comparable example arises, that is fine.

Function Name: **countDownByTwos**

Parameters:

start - an integer greater than 0 representing the starting number of the countdown

Return Value:

None

Test Cases:

```
 >>> countDownByTwos(5)
5
3
1
Blast Off!

>>> countDownByTwos(1)
1
Blast Off!

>>>countDownByTwos(6)
6
4
2
Blast off!
```

Description:

Write a function to count down from a given number by twos. The function should **print** the numbers from the given number to 1 (decreasing by 2 each time) in descending order, with each number being printed on its own line. After printing the required numbers, on a separate line, print the string 'Blast off!'

Function Name: **multiplicationTables**

Parameters:

number – an integer representing the number for which you want to create a multiplication table

limit – an integer representing how high you want the multiplication table to go

Return Value:

None

Test Cases:

```
>>>multiplicationTables(3, 4)
3*0 = 0
3*1 = 3
3*2 = 6
3*3 = 9
3*4 = 12

>>>mutiplicationTables(5, 2)
5*0 = 0
```

5*1 = 5
5*2 = 10
Description:
Write a function that takes in a two numbers. The first number is the number for which you wish to make a multiplication table; the second is how far you want the table to go. Have your function print out lines of the multiplication table as shown in the test case, by **printing** the number, the multiplication sign, the number you are multiplying it by, the equal sign, and what they equal. Note that the number*number do not have spaces between them, while the space-equal-number does!

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 + “.”
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.”
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.

Function Name: **comboLock**

Parameters:

num1 – a positive integer representing the first digit in the combination
num2 – a positive integer representing the second digit in the combination
num3 – a positive integer representing the third digit in the combination
num4 – a positive integer representing the fourth digit in the combination
num5 – a positive integer representing the fifth digit in the combination

Return Value:

Either the string “You opened the lock.” or the string “You are locked out.”

Test Cases:
1. comboLock(8, 2, 5, 4, 2) --> “You are locked out.”
2. comboLock(2, 8, 3, 6, 7) --> “You opened the lock.”
3. comboLock(11, 2, 5, 6, 4) --> “You are locked out.”
Description:
You own a combination lock that only opens when presented with the correct sequence of odd and even numbers that are less than 10. Write a function that takes in 5 integers.

Check whether they are in this order: even, even, odd, even, odd. If they are in the correct order and all below 10, then **return** the string “You opened the lock.” Otherwise, return “You are locked out.”

Function Name: **badRecord**

Parameters:

sentence - a string with at least one character

Return Value:

A string containing the capital letter characters from the input string.

Test Cases:

1. badRecord("CS is fun! I love coding.") --> "CSI"
2. badRecord("My Favorite Food is Pizza.") --> "MFFP"
3. badRecord("oooooO") --> "O"

Description:

Write a function that uses a for loop to create and **return** a new string that contains the capital letters the original input string. You may use a for loop to automatically index into the sequence. If the input string has no capital letters, you must return an empty string.

Function Name: **printTimestable**

Parameters:

none

Return Value:

none

You are hired to develop an educational software package. Your first job: Write a function printTimestable() that will *print* the times tables (up to 9) on the screen. When your function is called, it should print the following:

Times:	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Note that your function must print a header (Times: 1...9) and a first column number that goes from 1..9, 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 to space your grid out correctly.

Function Name: **printTimes**

Parameters:

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

Return Values:

none

Your boss was impressed with your 9x9 times table function. Now he wants you to modify the function so that it will work for for any sized times table. Write a printTimes(N) function that will print a times table from 1 up to N, for any positive number N.

Grading

You will earn points as follows for each function that works correctly according to the specifications.

checkHeight	**5**
function takes in a height in centimeters	2
function returns correct output for all valid inputs	3
smallestAndLargest	**10**
function accepts 3 parameters as floating point numbers	5
function returns correct output for all valid inputs	5
countDownByTwos	**5**
function prints numbers starting at specified parameter	2
function print decreases by 2 every time	2
function stops printing at 1 and ends with "blastoff"	1
multiplicationTables	**10**
function accepts two parameters	2
function correctly generates multiplication tables	5
function displays the multiplication table in proper format	3
complimentMaker	**10**
function accepts parameters as booleans	4
function correctly generates string output	6
comboLock	**15**
correctly accepts five integer parameters	5
correctly displays “You opened the lock.” when appropriate	5
correctly displays “You are locked out.” when appropriate	5
badRecord	**15**
uses a for loop	5
does not use indexing or slicing	5
returns correct output for any valid input	5

printTimestable	**10**
function prints correct multiplication output	5
function prints the correct multiplication with correct formatting	5
printTimes	**20**
function accepts an integer *n* as a paremeter	5
function correctly prints *n x n* times table	5
function nicely formats the output	5
function does not return any value	5