CS 2316

Individual Homework 2 – Conditionals & Loops

Due: Thursday, May 26th, before 11:55 PM

Out of 100 points

Files to submit: **1. HW2.py**

This is an INDIVIDUAL assignment!

Collaboration at a reasonable level will not result in substantially similar code. 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. You should not exchange code or write code for others.

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.

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.

1. checkAge
2. largest
3. countOdds
4. carDescription
5. divisionTables
6. comboLock
7. badRecord
8. printTimes
9. encode

Function Name: **checkAge**

Parameters:

rating – a string which will either be “G”, “PG”, “PG-13”, or “R”
age – an integer representing the age of the patron

Return Value:

Either the string “Enjoy the movie!” or the string “Sorry, you’re not old enough to see this movie.”

Test Cases:

checkAge(“G”, 10) --> “Enjoy the movie!”
checkAge(“PG”, 7) --> “Enjoy the movie!”
checkAge(“PG-13”, 10) --> “Sorry, you’re not old enough to see this movie.”
checkAge(“PG-13”, 13) --> “Enjoy the movie!”
checkAge(“R”, 13) --> “Sorry, you’re not old enough to see this movie.”
checkAge(“R”, 17) --> “Enjoy the movie!”

Description:

Write a function for the local movie theater that will determine whether a particular patron is old enough to see a particular movie. The function will allow a patron of any age to see a movie rated “G” or “PG”. Anyone who is 13 years or older may see a movie rated “PG-13”, and anyone who is 17 years old or older may see a movie rated R. If a patron wishes to see a movie for which they are not old enough, your function should **return** “Sorry, you’re not old enough to see the movie.” If they are old enough, you should **return**, “Enjoy the movie!”

Function Name: **largest**

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 floating point number that represents the largest number of the three given values.

Test Cases:

smallest(433.1, 2340.32, 12323.7) --> 12323.7
smallest(12.0, 32.1, 32.1) --> 32.1
smallest(23.44, 23.44, 23.44) --> 23.44

Description:

Write a function that takes in three numbers as parameters and **returns** the largest of the three. If two or all of the numbers are equal, then just return one of them. The function may not use any built-in math functions, e.g. you may not use max().

Function Name: **countOdds**

Parameters:

start - an integer greater than 0 representing the last number of the count sequence

Return Value:

None

Test Cases:

1. >>> countOdds(4)
1
3
I made it!

2. >>> countOdds(7)
1
3
5
7
I made it!

Description:

Write a function to count every odd number from one to a given number (inclusive). The function should **print** the numbers from 1 up to the given number in ascending order, with each number being printed on its own line. If the given number is odd, it should be included in the count. If the given number is even, the last odd number before it should be the ending point. After printing the required numbers, on a separate line, **print** the string ' I made it!'

Function Name: **carDescription**

Parameters:

color – a string representing the color of the car
age - a boolean (True or False) representing whether the car is "new" (T) or “old” (F)
answer1 - a boolean (True or False) representing whether the car is "awesome”
answer2 - a boolean (True or False) representing whether the car is “expensive”
answer3 - a boolean (True or False) representing whether the car is “small”

Return Value:

The string “That ” + age + color + “car is ” + the designated compliments + “.”

Test Cases:

1. carDescription(“gold”, True, True, True, True) --> “That new gold car is awesome expensive small.”
2. CarDescription(“blue”, True, False, True, False) --> “That new blue car is expensive. ”
3. CarDescription(“red”, False, False, False, False) --> “That old red car is unnoticeable.”

Description:

Write a function that returns a string of descriptions based on the adjectives selected by the inputs. Use a string representing a single color for the first parameter, and the inputs True and False for all booleans. The function should return the string “That ” concatenated with the “new” if True or “old” if False, the color string, “car is “, and the final compliments that are true. The three final compliments should be: "awesome" "expensive" and "small". If none of the final compliments are true, add the description “unnoticeable”.

Function Name: **divisionTables**

Parameters:

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

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

Return Value:

None

Test Cases:

1. >>>divisionTables(3, 4)
3 / 1 = 3.00
3 / 2 = 1.50
3 / 3 = 1.00
3 / 4 = 0.75

2. >>>divisionTables(2, 5)
2 / 1 = 2.00
2 / 2 = 1.00
2 / 3 = 0.67
2 / 4 = 0.50
2 / 5 = 0.40

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 division table as shown in the test case, by printing the number, the division sign, the number you are multiplying it by, the equal sign, and what they equal. Note the number after the equal signs is a float printed to only the second decimal place!

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(12, 2, 5, 6, 3) --> “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 only the lowercase letters from the input string.

Test Cases:

1. badRecord(“CS is fun! I love coding.”) --> “ isfunlovecoding”
2. badRecord(“My Favorite Food is Pizza.”) --> “yavoriteoodisizza”
3. badRecord(“oooooO!!!!”) --> “ooooo”

Description:

Write a function that uses a for loop to create and return a new string that contains only the lowercase letters the original input string. You may use a for loop to automatically index into the sequence. If the input string has no lowercase letters, you must return an empty string. *Hint: The string module has a constant which lists all of the lowercase ascii letters. To use it, you must import the string module, which allows you to refer to string.ascii_lowercase.*

```
>>> import string
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
```

Function Name: **printTimes**

Parameters:

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

Return Value:

none

You are hired to develop an educational software package. Your first job: Write a function printTimes(N) that will print the times tables (1 to N) on the screen for any positive N. When your function is called with N = 9 for example, 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...N) and a first column number that goes from 1..N, 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: **encode**

Parameters:

string1 - a string with at least one character
string2 - a string with at least one character

Return Values:

A string containing the encoded words

Test Cases:

1. encode(“Hello”, “World”) --> “HWeolrllod”
2. encode(“CS”, “coding”) --> “CcSoding”
3. encode(“almost”, “done”) --> “adlomnoest”

Description

Your boss has recently found out a competing company has been stealing your companies’ ideas! Because he was impressed with your times table function, your boss has asked you to fix this new problem. Write a function named encode that uses a loop to combine string1 and string2 by alternating characters from each parameter and return the final output. The loop should pick a character from string1 first and then string2 until one of the strings has no more letters. When one string runs out of letters, the rest of the other string should be added to the end.

Grading

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

checkAge	5
smallest	5
countOdds	5
carDescription	10
divisionTables	10
comboLock	10
badRecord	15
printTimes	20
encode	20

For 3 points of extra credit, make the output of the carDescription function have correct English grammar in all cases. (correct placement of commas and "and" before the last of multiple adjectives for any number of adjectives).