CS 1301

Individual Homework 3 – Conditionals & Loops

Out of 100 points

Due Friday September 14th before 11:55pm

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 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.

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 name, section, GTID/Email, and your collaboration statement. Also include each of the following functions.

1. rideHeight
2. sandwichMaker
3. factorial
4. wordMirror
5. stringChange
6. onlyLetters
7. safeCrack
8. arrowHead
9. printTimes

Function Name: **rideHeight**

Parameters:

An Integer.

Return:

A String.

Test Cases:

rideHeight(122) --> “Sorry. You must be at least 1 meter 28 cm to ride.”
rideHeight(170) --> “Have a safe flight!”
rideHeight(128) --> “Have a safe flight!”

Description:

Write a function for the Space Mountain ride at Disney World that determines whether the user is at least 1 meter 28 centimeters tall so that he or she can ride the roller coaster. If the user's height, which is provided in centimeters by the parameter, is greater than or equal to the minimum height, **return** the string 'Have a safe flight!'. Otherwise, **return** the string 'Sorry. You must be at least 1 meter 28 cm to ride.'

Function Name: **sandwichMaker**

Parameters:

A Boolean (True or False) representing whether the sandwich has "bacon".
A Boolean (True or False) representing whether the sandwich has "lettuce".
A Boolean (True or False) representing whether the sandwich has "tomato".
A Boolean (True or False) representing whether the sandwich has "mayo".

Retur

A String.

Test Cases:

sandwichMaker(True, True, True, False) --> “The sandwich has bacon lettuce tomato."
sandwichMaker(False, True, False, True) --> “The sandwich has lettuce mayo.”
sandwichMaker(False, False, False, False) --> “Not Hungry?"

Description:

Write a function that **returns** a string of toppings based on the adjectives selected by the parameters. The function will accept four Boolean parameters (True or False). The function should return the string “The sandwich has ” concatenated with the toppings on the sandwich. The four toppings should be:"bacon", "lettuce", "tomato", and "mayo", corresponding respectively to the four parameters.
If none of the parameters are True, return the string “Not Hungry?"

Function Name: **factorial**

Parameters:

A positive Integer (not a Float).

Return:

An Integer

Test Cases:

factorial(4) --> 24
factorial(5) --> 120
factorial(0) --> 1

Description:

Your function should take in a number and compute the factorial, **using a for loop or a while loop**. An example of the mathematical formula for a factorial is: 5! or “5 factorial" which equals 5*4*3*2*1”, which equals 120. As a reminder, 0! is equal to 1. **DO NOT** use a built in factorial function for this problem.

Function Name: **wordMirror**

Parameters:

A String.

Return:

A String.

Test Cases:

wordMirror("CS1301") --> “CS1301|1031SC”
wordMirror("Hello World!") --> “Hello World!|!dlroW olleH”
wordMirror("racecar") --> “racecar|racecar”

Description:

Write a function that takes in a string and **returns** the string followed by a **vertical line** (|) and then the original string in reverse. Hint: Use a while loop or for loop to get the original string in reverse, then build the final string from there.

Function Name: **stringChange**

Parameters:

A String.

Return:

A String.

Test Cases:

stringChange(“I can't swim!") --> “I CAN NOT SWIM!"
stringChange(“I don't like squash.”) --> “I do not like squash.”
stringChange(“I shouldn't procrastinate.”) --> “I should not procrastinate.”

Description:

Write a function that creates and **return** a new string that contains no contractions. Your code should recognize **FOUR** contractions:
can't -> can not ; won't -> will not ; don't -> do not ; shouldn't -> should not
Futhermore if the string ENDS with an **exclamation mark** return the string in all capital letters.

Function Name: **onlyLetters**

Parameters:

A String.

Return:

A String.

Test Cases:

onlyLetters("gburdell3") --> “gburdell”
onlyLetters("Hello@World.com") --> “HelloWorldcom”
onlyLetters("2012") --> “”

Description:

Write a function that uses a for loop to create and **return** a new string that contains only the letters of the original input. If the input string has no letters, you must return an empty string.

Function Name: **safeCrack**

Parameters:

A positive Integer representing the first digit in the combination.
A positive Integer representing the second digit in the combination.
A positive Integer representing the third digit in the combination.
A positive Integer representing the fourth digit in the combination.
A positive Integer representing the fifth digit in the combination.

Return:

A String.

Test Cases:

safeCrack(6, 1, 6, 3, 4) --> “You are locked out.”
safeCrack(8, 3, 4, 5, 2) --> “You opened the lock.”
safeCrack(8, 5, 14, 3, 6) --> “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, odd, even, odd, even. Furthermore the first number, **must** be an **8.** If they are in the correct order, the first number is an 8, and all below 10, then **return** the string “You opened the lock.” Otherwise, return “You are locked out.”

Function Name: **arrowHead**

Parameters:

An odd Integer between 3 and 61, inclusive.

Return:

None.

Test Cases:

```
>>> arrowHead(3)
  *
***
  *
  *
  *
>>> arrowHead(5)
    *
  ***
*****
    *
    *
    *
>>> arrowHead(7)
      *
    ***
  *****
*******
      *
      *
      *
>>>
```

Description:

Your function will draw an arrowhead on-screen using the print function. The parameter represents the number of Asterisks in the bottom level of the arrowhead. Your arrowhead will have one Asterisk character at the top level, three at the 2nd to top level, five at the 3rd level, and so on, until it reaches the bottom level. After the bottom level you should have three lines with a single Asterisk to represent the trunk. Note that your arrowhead is NOT centered on the screen. The left hand side of the base of the tree will line up with the left hand side of the python window. You will have to figure out for yourself how many spaces to leave for the 1st and subsequent levels of the arrowhead so that everything works out right!

Function Name: **printTimes**

Parameters:

A positive Integer.

Return:

None.

Description:

You are hired to develop an educational software package. Your first job: Write a function printTimes() that will ***print*** the times tables (1 to N, inclusive) on the screen.

```
>>> printTimes(9)
```

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

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 to space your grid out correctly.

Grading

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

rideHeight	**5**
function takes in a height in centimeters	2
function returns correct output for all valid inputs	3
sandwichMaker	**10**
function accepts parameters as booleans	4
function correctly generates string output	6
factorial	1**0**
function loops through numbers	3
function returns the correct integer value	7
wordMirror	**5**
function uses a loop to traverse through the string	2
function correctly generates string output	3
stringChange	**15**
uses loops or appropriate string methods	7
returns correct output for any valid input	8
onlyLetters	**10**
uses a for loop	4
returns correct output for any valid input	6
safeCrack	**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
arrowHead	**10**
Function prints asterisks in a arrowHead shape	5
function prints correct number of asterisks / lines	5
printTimes	**20**
function accepts an integer *n* as a paremeter	5
function correctly prints *n x n* times table	10
function nicely formats the output	5