CS 1301

Individual Homework 3 – Conditionals & Loops

Out of 100 points

Due Monday February 6th 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. checkHeight
2. complimentMaker
3. isPrime
4. wordReverse
5. badRecord
6. allLetters
7. comboLock
8. xmasTree
9. printTimes

Function Name: **checkHeight**

Parameters:

An Integer.

Return:

A String.

Test Cases:

checkHeight(130) --> “Sorry. You must be at least 1 meter 32 cm to ride.”
checkHeight(137) --> “Have a great ride!”
checkHeight(170) --> “Have a great ride!”

Description:

Write a function for the Goliath ride at Six Flags that determines whether the user is at least 1 meter 32 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 great ride!'. Otherwise, **return** the string 'Sorry. You must be at least 1 meter 32 cm to ride.'

Function Name: **complimentMaker**

Parameters:

A Boolean (True or False) representing whether the user is "awesome".
A Boolean (True or False) representing whether the user is "super".
A Boolean (True or False) representing whether the user is "smart".
A Boolean (True or False) representing whether the user is "cool".

Return:

A String.

Test Cases:

complimentMaker(True, True, True, True) --> “You are awesome super smart cool.”
complimentMaker(False, True, False, True) --> “You are super cool.”
complimentMaker(False, False, False, False) --> “No Comment.”

Description:

Write a function that **returns** a string of compliments based on the adjectives selected by the parameters. The function will accept four Boolean parameters (True or False). The function should return the string “You are ” concatenated with the compliments that are true. The four compliments should be: "awesome", "super", "smart", and "cool", corresponding respectively to the four parameters. If none of the parameters are True, return the string “No Comment.”

Function Name: **isPrime**

Parameters:

A positive Integer (not a Float).

Return:

A Boolean.

Test Cases:

isPrime(2) --> True
isPrime(1993) --> True
isPrime(100) --> False

Description:

Your function should take in a number and check if the value is a prime number, using a for loop or a while loop. If the number is a prime, **return** True. If it is not a prime, **return** False.

Function Name: **wordReverse**

Parameters:

A String.

Return:

A String.

Test Cases:

wordReverse("Computer Science") --> “ecneicS retupmoC”
wordReverse("Hello World!") --> “!dlroW olleH”
wordReverse("racecar") --> “racecar”

Description:

Write a function that uses a while loop to traverse through the string and **return** the string backwards.

Function Name: **badRecord**

Parameters:

A String.

Return:

A String.

Test Cases:

badRecord(“CS is fun! I love coding.”) --> “CSI”
badRecord(“My Favorite Food is Pizza.”) --> “MFFP”
badRecord(“oooooO”) --> “O”

Description:

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

Function Name: **allLetters**

Parameters:

A String.

Return:

A String.

Test Cases:

allLetters("gburdell3") --> “gburdell”
allLetters("Hello@World.com") --> “HelloWorldcom”
allLetters("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: **comboLock**

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:

comboLock(7, 2, 5, 4, 2) --> “You are locked out.”
comboLock(3, 8, 3, 6, 7) --> “You opened the lock.”
comboLock(11, 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: odd, 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: **xmasTree**

Parameters:

An odd Integer between 3 and 61, inclusive.

Return:

None.

Test Cases:

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

Description:

Your function will draw a Christmas tree on-screen using the print function. The parameter represents the number of Asterisks in the bottom level of the Christmas tree. Your Christmas tree 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 Christmas tree 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 Christmas tree 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.

checkHeight	**5**
function takes in a height in centimeters	2
function returns correct output for all valid inputs	3
complimentMaker	**10**
function accepts parameters as booleans	4
function correctly generates string output	6
isPrime	1**0**
function loops through numbers	3
function returns the correct Boolean value	7
wordReverse	**5**
function uses a while loop to traverse through the string	2
function correctly generates string output	3
badRecord	**15**
uses a for loop	7
returns correct output for any valid input	8
allLetters	**10**
uses a for loop	4
returns correct output for any valid input	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
xmasTree	**10**
Function prints asterisks in a xmasTree 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