CS 1301 Individual Homework 3 – Conditionals & Loops

Due: **Friday, Sep 12th, 2014 before 11:55 pm**
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: **tallEnough (5 pts)**

Parameters:

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

Return Value:

True or False (as a Boolean)

Test Case:

tallEnough(56) returns True

tallEnough(12) returns False

tallEnough(75) returns False

Description:

Your friend Bob is studying abroad in Europe and he goes to Disneyland Paris one weekend. There is this ride he wants to go on but it says, “You have to be taller than 120 cm and shorter than 190 cm to go on this ride”. Unfortunately Bob doesn’t know his height in centimeters. You should write a function to help your friend out. This function should take his height in inches as a parameter (type: integer) and

should convert it to meters using this formula 1 cm = 1 in / 0.39370. Then it should check and see if your friend Bob can go on the ride. If he can, (if his height is more than 120 cm, and less than 190 cm) then the function should return True, otherwise it should return False.

Function Name: **whereIsWaldo (5 pts)**

Parameters:

Int1- the x coordinate of Waldo's position

Int2- the y coordinate of Waldo's position

Return Value:

Either a string saying "You found Waldo" or "Couldn't find Waldo. Better luck next time"

Test Cases:

whereIsWaldo(5,4)

- Input Box: "Guess Waldo's x-coordinate": 5
- Input Box: "Guess Waldo's y-coordinate": 4
- You found Waldo

whereIsWaldo(1,4)

- Input Box: "Guess Waldo's x-coordinate": 5
- Input Box: "Guess Waldo's y-coordinate": 4
- Couldn't find Waldo. Better luck next time

Description:

Write a function to do a fun game of where is Waldo?. The function should take in two parameters: the x-coordinate and the y-coordinate of Waldo's position. Then, the function asks for the user to input guesses separately for x and y coordinates. If the combination of these two is right, then it should return a string that says "You found Waldo" and if it

is wrong, the string that is returned should say “Couldn’t find Waldo. Better luck next time.”

Function Name: **allLetters (10 pts)**

Parameters:
userString - 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.
You MUST use a for 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.

Function Name: **replaceLetter (15 pts)**

Parameters:
aString- string that the user enters
aLetter- the letter that the user enters
Return Value:
None
Test Cases:
replaceLetter(“Hello”, “a”)

 Input Box: “Input a letter” – b
 Input Box: “Letter not in string. Input a letter “– o
 Hella.

replaceLetter(“I love CS 1301”, “p”)
 Input Box :“Input a letter” – o
 I lpve CS 1301

Description:

Write a function that takes two parameters as shown above. The function should ask the user to input a letter (using the input box), which might or might not be in the string. If the letter is in the string, the function then goes ahead and replaces the letter that the user put from the input box with the letter that was passed as a parameter and prints out the new word. If the letter that the user inputs is not in the string, the function should print “Letter not in string” and should ask the user to input another letter.

Function Name: **countUp (15 pts)**

Parameters:
start- an integer marking the starting number
end- an integer marking the ending number
increment- an integer marking the increment

Return Value:
None

Test Cases:
CountUp(5,9,2)
5
7

9
Done!

CountUp(6,10,1)
6
7
8
9
10
Done

CountUp(1,5,4)
1
5
Done!
Description:

Write a function that takes in three parameters, a starting number, an ending number and the increment. The function MUST use a while loop to print starting from the starting number all the way to the ending number all in a new line using the increment. After the ending number is reached the function should print "Done!" in a new line. The function should return None. You can assume that the user will always put a starting number smaller than the ending number, and the ending number could be reached by the increment by the starting number.

Function Name: **numMountainRange (20pts)**

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

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.

Test Cases:

You have X number of rows, but note that there are two 1s, four 2s, six 3s, eight 4s, etc.

```
python>>> numMountainRange(2)
1  1
2222
Ok
python>>> numMountainRange(4)
1       1
22     22
333   333
44444444
Ok
python>>> numMountainRange(9)
1                   1
22                 22
333               333
4444             4444
55555           55555
666666         666666
7777777       7777777
88888888     88888888
999999999999999999
Ok
```

Function Name: **printTimestable (10 pts)**

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 (20 pts)**

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.

For example, when your function is called as printTimes(4) , it should print the following:

Times: 1 2 3 4
1 1 2 3 4

2 2 4 6 8
3 3 6 9 12
4 4 8 12 16

Grading Rubric

tallEnough **5pts**

- function takes in a height in inches 1

- function correctly converts to centimeter 2
- function returns either True or False correctly 2

whereIsWaldo **5pts**

- function asks the user to input x and y coordinates 2
- function should take two parameters 1
- function returns the correct string 2

allLetters **10 pts**

- uses a for loop 4
- returns correct output for any valid input 6

replaceLetter **15pts**

- function accepts three parameters 3
- function asks the user to input a letter 2
- the input box pops again if letter not in string 5
- the function correctly replaces letter 3
- the function prints out the new word 2

countUp **15pts**

- function accepts three parameters as integers 3
- function uses a while loop 5
- function prints out the right results 5
- function prints out “Done!” at the end 2

numMountainRange **20pts**

- Correct number of rows and correct number in rows 10
- Correct shape (-5 if hard coded) 10

printTimestable **10pts**

- function prints correct multiplication output 5
- function prints with correct formatting 5

printTimes **20pts**

- 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

Elements of this homework created by Gunce Yalcin