CMSC 201 Spring 2019

Homework 6 – Recursion & File I/O

Due Date: Friday, April 26th, 2018 by 11:59:59 PM
Value: 40 points

Collaboration: For Homework 6, collaboration is allowed. Make sure to consult the syllabus about the details of what is and is not allowed when collaborating. You may not work with any students who are not taking CMSC 201 this semester.

If you work with someone, remember to note their name, email address, and what you collaborated on by filling out the Collaboration Log.

You can find the Collaboration Log at http://tinyurl.com/spring19-201-collab.

Remember that all collaborators need to fill out the log each time; even if the help was only "one way" help.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:     FILENAME.py
# Author:   YOUR NAME
# Date:     THE DATE
# Section:  YOUR DISCUSSION SECTION NUMBER
# E-mail:   YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Instructions

For each of the questions below, you are given a problem that you must solve or a task you must complete.

Objective

Homework 6 is designed to give you lots of practice with recursion. You should think carefully about the base case(s), recursive case(s), and recursive call(s) that each problem will need.

If a solution does not use recursion, you will earn zero points for that question, even if your code solves the problem.

You may not use `for` or `while` loops for these problems unless otherwise specified!

Coding Standards

Prior to this assignment, **you should be familiar with the entirety of the Coding Standards**, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

You should be commenting your code, and using constants in your code (not magic numbers or strings).
Any numbers other than 0 or 1 are magic numbers!

You will **lose major points** if you do not follow the 201 coding standards.

If you have questions about commenting, whitespace, or any other coding standards, please come to office hours.

Additional Instructions – Creating the hw6 Directory

Just as you did for previous homeworks, you should create a directory to store your Homework 6 files. We recommend calling it **`hw6`**, and creating it inside the **`Homeworks`** directory inside the **`201`** directory.

Questions

Each question is worth the indicated number of points. Following the coding standards is worth 4 points. Failing to have complete file headers or failing to have correctly named files will lead to a deduction of points.

hw6_part1.py **(Worth 9 points)**

For this part, you will be writing a function, `isFaultyPalindrome()`, that returns True if a string is a palindrome with an acceptable number of faults. Your function should be case sensitive.

A palindrome is a string that reads the same way backwards and forwards, i.e. the second half is a mirror of the first half. A fault is a character that is not mirrored in the other half of the string.

You can assume that the user is going to input a string and a non-negative integer.

Your isFaultyPalindrome implementation MUST be recursive and MUST use the two following parameters.

```
#########################################################
# isFaultyPalindrome() determines if a string is a
#   palindrome
# Input:    word; a string that this function will assess
#           faultsPermitted; an integer representing the
#           maximum tolerated faults in word
# Output:   True if word is a palindrome with at most
#           faultsPermitted faults, false otherwise
```

Sample output for this problem can be found on the next page.

Here is some sample output for hw6_part1.py, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux[0]$ python hw6_part1.py
Input a word:aba
Input number of permitted faults:0
Looks good!

linux[0]$ python hw6_part1.py
Input a word:tacocat
Input number of permitted faults:0
Looks good!

linux[0]$ python hw6_part1.py
Input a word:1234567890987654321
Input number of permitted faults:0
Looks good!

linux[0]$ python hw6_part1.py
Input a word:1234567890987655555
Input number of permitted faults:4
Looks good!

linux[0]$ python hw6_part1.py
Input a word:1234567890987655555
Input number of permitted faults:3
That's really not a palindrome.

linux[0]$ python hw6_part1.py
Input a word:abcdcba
Input number of permitted faults:2
Looks good!

TODO: add an example with case causing it to fail
```

w6_part2.py **(Worth 9 points)**

You must write a function **`alternateCharacters()`** that weaves two strings together one character at a time, and returns the resulting string.

You can assume that the user will input two strings.

REMINDER: You are not permitted to use any kind of loop to complete this or other parts of the homework.

```
#################################################################
# alternateCharacters(): Given two strings, return a string that
# alternates characters between the two strings
# Input:   phrase1: the first string
#          phrase2: the second string
# Output:  a string that alternates characters between phrase1 and
#          phrase2
```

Sample output for this problem can be found on the next page.

Here is some sample output for hw6_part2.py, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux[0]$ python hw6_part2.py
Say something: abcd
Say something else: 1234
a1b2c3d4

linux[0]$ python hw6_part2.py
Say something: How do you do, fellow children?
Say something else: What is up, other humans?
HWohwa td oi sy ouup ,d oo,t hfeerl lhouwm acnhsi?ldren?

linux[0]$ python hw6_part2.py
Say something: Would you care to join me for a rousing
game of Othello?
Say something else: No. Just no.
WNoou.l dJ uysotu  ncoa.re to join me for a rousing game
of Othello?

linux[0]$ python hw6_part2.py
Say something: Why not?
Say something else: You really really really really
wouldn't understand.
WYhoyu  nroeta?lly really really really wouldn't
understand.
```

hw6_part3.py (Worth 9 points)

For this part, you will implement modulo arithmetic recursively. You must implement a function `recursiveMod()` which takes TWO INTEGERS ONLY and does not contain `for` or `while loops`.

The inputs from the user are guaranteed to be positive integers.

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

```
linux[0]$ python hw6_part3.py
Enter a number: 30
Enter another number: 7
30 % 7 = 2

linux[0]$ python hw6_part3.py
Enter a number: 2
Enter another number: 10
2 % 10 = 2

linux[0]$ python hw6_part3.py
Enter a number: 10
Enter another number: 7
10 % 7 = 3
```

hw6_part4.py (Worth 9 points)

Next, you will create a program that draws a right triangle, using a height and symbols provided by the user.

The program must contain a `main()` and a recursive function called **`recursiveTri()`**. The program may also contain any other functions you deem necessary. for and while loops are not permitted.

For these inputs, you can assume the following:

- The height will be a positive integer (zero or greater)
- The symbols will each be a single character

Here is some sample output, with the user input in **blue**.
(Yours does not have to match this word for word, but it should be similar.)

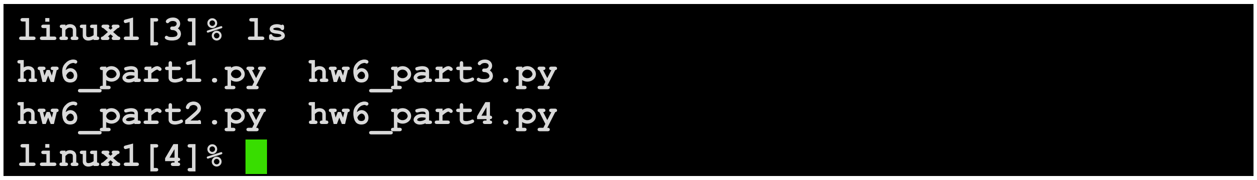
```
linux[0]$ python hw6_part4.py
Please enter the height of the triangle:3
Please enter the outline symbol to use:Y
Please enter the fill symbol to use:&
YYY
YY
Y

linux[0]$ python hw6_part4.py
Please enter the height of the triangle:10
Please enter the outline symbol to use:V
Please enter the fill symbol to use:@
VVVVVVVVVV
V@@@@@@@@V
V@@@@@@@V
V@@@@@@V
V@@@@@V
V@@@@V
V@@@V
V@@V
V@V
VV
V
```

Submitting

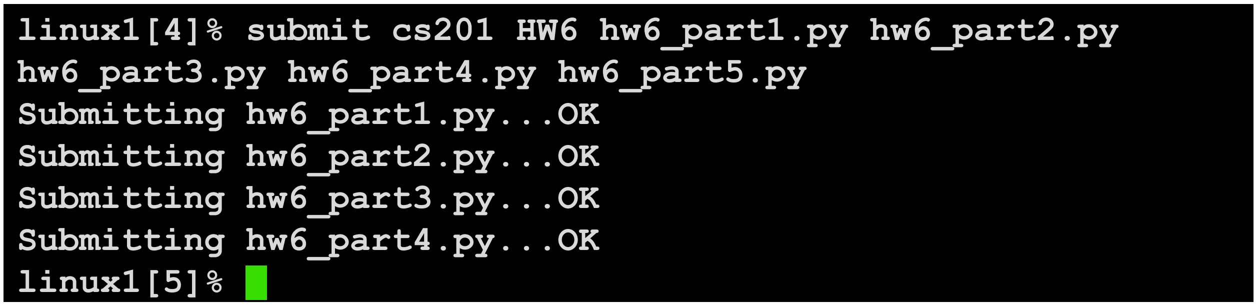
Once your **hw6_part1.py**, **hw6_part2.py**, **hw6_part3.py** and **hw6_part4.py** files are complete, it is time to turn them in with the **submit** command. (You may also turn in individual files as you complete them. To do so, only **submit** those files that are complete.)

You must be logged into your account on GL, and you must be in the same directory as your Homework 6 Python files. To double-check you are in the correct directory with the correct files, you can type **ls**.



```
linux1[3]% ls
hw6_part1.py  hw6_part3.py
hw6_part2.py  hw6_part4.py
linux1[4]%
```

To submit your Homework 6 Python files, we use the **submit** command, where the class is **cs201**, and the assignment is **HW6**. Type in (all on one line) **submit cs201 HW6 hw6_part1.py hw6_part2.py hw6_part3.py hw6_part4.py** and press enter.



```
linux1[4]% submit cs201 HW6 hw6_part1.py hw6_part2.py
hw6_part3.py hw6_part4.py hw6_part5.py
Submitting hw6_part1.py...OK
Submitting hw6_part2.py...OK
Submitting hw6_part3.py...OK
Submitting hw6_part4.py...OK
linux1[5]%
```

If you don’t get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**