Version: 2/7/2013 (10:30 PM) **Homework Due Wednesday 2/13/2013**

HW5: CS 110X C 2013

Note: This homework (and all remaining homework assignments) is a **partner homework** and must be completed by each partner pair. When you complete this assignment, you must not share your answers with any other student. Only one person from a partner pair needs to submit the assignment.

The goal of this homework is to prepare you for the second Examination. It consists of a number of Skills Areas covered in class. Each of these problems is independent of the others, and you can solve them in any order.

Please make sure that **as a team** you work together on these problems, but you also each individually understand the code for each of the associated programs.

As I stated in class, completing this homework is the best preparation you can have for the exam coming up on Tuesday February 12th.

File Input and Output Skills

IO-3.	Know how to write data to a file
IO-4.	Know how to read in text data containing strings of text
IO-5.	Know how to read in CSV formatted data

Q1	File Manipulation

Write a python function `applyFilter(inputFileName, outputFileName, filterString)` that takes an `inputFileName` string, an `outputFileName` string, and a `filterString`.

Once execution completes, a file is created on disk in the current working directory whose name is `outputFileName`. The contents of this file are all lines from the file identified by `inputFileName` that contained the given `filterString`. Be sure to close all files that you opened. Your function must return the count of the number of lines written to `outputFileName`.

Sample Output

```
>>> applyFilter('in.txt', 'out.txt', 'T')
2
>>> applyFilter('in.txt', 'out.txt', 'S')
0
```

in.txt	out.txt	filter
This Is A Test	This Test	T

Q2 Processing CSV files

Write a python function `sumRows(inputFileName)`. This function reads in a file containing lines of the form `"val1,val2,val3,…"` where each value is an integer. Your function will then return a list containing the sum total of the values in each row. You can assume that each line has at least a single value on it and that the input file has at least one line contained within it.

Sample Output

```
>>> sumRows('numbers.txt')
[16, 485, 271, 22, 307, 12, 69]
```

The above output results from invoking `sumRows` on the `'numbers.txt'` file on the right. Be sure to close all files that you opened.

numbers.txt

```
1,2,3,4,6
234,234,12,5
234,34,2,1
21,1
23,23,4,23,234
12
1,21,12,32,3
```

Python Functions

PF-2.	Know how to define a function with parameters
PF-3.	Know how to use **return** statement within a function
CS-10.	Understand nesting of **for** and **while** loops

Q3 Passing values between functions (parameters and return values)

Write a `shuffle (list1, list2)` function that takes two equal-length lists containing values and returns a new list that contains alternating elements from these two lists.

You can assume that both lists have at least one element in them.

Sample Output

```
>>> shuffle(['a', 'b', 'c'], [1,2,3])
['a', 1, 'b', 2, 'c', 3]
>>> shuffle([1,2,3],[9,8,7])
[1, 9, 2, 8, 3, 7]
```

Q4 Using for loops

Write a Python function `explosion(stringList, multipleList)` that takes two equal-length lists as input and returns a string value. `stringList` contains a list of string values; `multipleList` contains a list of integers. The string value returned by this function is the concatenation of each element in `stringList` multiple times, based upon the corresponding value in `multipleList`. You can assume all integers in `multipleList > 0` and that `stringList` has at least one element.

Sample output appears on the right.

Sample Output

```
>>> explosion(['the'], [3])
'thethethe'
>>> explosion(['the', 'and'], [3, 2])
'thethetheandand'
```

Q5 More complicated looping

Write a Python function `isPermutation(base, target)` that takes two lists of values and determines (True or False) whether `target` is a permutation of the values in `base`. You don't know how long these lists are. You don't know whether the values are integers, floats, or Boolean values. **You only know that no value is repeated in either `base` or `target`, and that neither list is empty.**

In addition, **your code must not make any changes to these two lists**. They must remain unaffected after being used as arguments to your function. Your code must return `True` or `False`.

Your code must be able to handle the cases that are shown in this sample output.

Sample Output

```
>>> isPermutation([1, 5, 'z'], ['z', 5, 1])
True
>>> isPermutation([1, 5], [3, 5, 1])
False
>>> isPermutation([1,5,4,6,8,2,3,9,7],
[9,8,7,6,5,4,3,2,1])
True
```

Q6 Nested For Loops (This is a challenge problem)

Define a function `sameConsecutive(list, n)` that determines whether `list` has `n` consecutive values that are the same. The only thing you know about `list` is that it is not empty. You only know that `n` is an integer > 1.

Sample output appears on the right.

Sample Output

```
>>> sameConsecutive([9,7,7,7,6], 3)
True
>>> sameConsecutive([1,2,3,4,4], 2)
True
>>> sameConsecutive([1,1,2,2,4], 3)
False
```

Control Structures

CS-3.	Know **elif** statement
CS-4.	Know how to nest **if** statements
CS-6.	Know how to use **while** as indefinite loop

Q7 Indefinite loops and **elif** statements

Define a function `exactChange(n)` that takes an integer `0 < n < 100` representing a number of cents in US Currency. This function shall **return** a list of four integers ≥ 0 that determines the appropriate change in US currency equivalent. The allowed coins are penny (1¢), nickel (5¢), dime (10¢), and quarter (25¢). The returned list contains four values representing the count of the four coins in increasing currency values.

Sample output appears on the right. For example, 19¢ equals 4 pennies, 1 nickel and 1 dime.

Sample Output

```
>>> exactChange(19)
[4, 1, 1, 0]
>>> exactChange(99)
[4, 0, 2, 3]
>>> exactChange(1)
[1, 0, 0, 0]
```

Q8 ASCII ART (this is a challenge problem)

Create a function `printX(n)` that prints out an “X” using asterisk characters over a square of “.” characters. You can assume `n` is an odd number > 3.

Sample output appears on the right.

Hint: Try to break this up into three parts:

(1) print the upper part of the X

(2) print the middle line which contains a single “” character in the exact middle*

(3) print the bottom part of the X

Parts (1) and (3) will require a loop

Sample Output

```
>>> printX(5)
*...*
.*.*.
..*..
.*.*.
*...*
>>> printX(7)
*.....*
.*...*.
..*.*..
...*...
..*.*..
.*...*.
*.....*
```

How To Get Started On This Assignment

A template HW5.py file is provided to you and a Rubric exists also.

Submit your HW5.py file using the web-based turnin system. As we have mentioned in class, only one of the team members needs to submit the assignment. But just make sure that something gets submitted!

Make sure that you don’t write any additional code to invoke these functions, since that gets in the way of the TAs grading the assignments. Good Luck!