CS1110 Lab 7: More Practice for A5 (Apr 5-6, 2016)

SOLUTIONS

First Name: _____ **Last Name**: _____ **NetID:** _____

The lab assignments are very important. Remember this: *The lab problems feed into the assignments and the assignments define what the exams are all about.*

Start *before* your lab meets.

We recommend spending an hour or two on the lab *before* coming to your section, so you can use your in-person time to ask questions most efficiently.

Also, this strategy of starting beforehand increases your chances of checking in at your lab section, which will probably take less time than waiting in line at consulting hours!

Getting credit

Complete all required blank boxes and lines on this handout. When you are finished, show your written answers to one of the CS 1110 lab staff in your section on April 5-6 or in any consulting hours up to and including April 11 (earlier days have shorter lines). The staff member will ask you a few questions to make sure you understand the material, and then swipe your Cornell ID card or directly make a notation in CMS to record your success. This physical piece of paper is yours to keep.

Getting set up

From the Lab webpage, download and unzip `Lab_7.zip` into a folder named (for example) `Lab_7`. In the command shell, navigate the file system so that this folder is THE CURRENT WORKING DIRECTORY. Review the slides through the March 24 lecture.

1 Operations with Lists

Assume that `x = ['abc','abcd','abcde','abcdef']`. Now fill out the table:

	Expression	I Think the Value Is	Python Says	Notes
1	`len(x)`			
2	`'a' in x`			
3	`x[2][1:3]`			
4	`x.count('a')`			
9	`x[2].split('c')`			
5	`x.extend(100)`			
8	`x.extend([100])`			

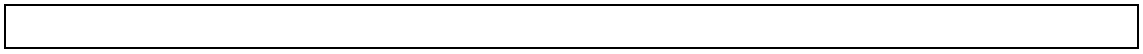
Solution:

```
>>> x = ['abc','abcd','abcde','abcdef']
>>> len(x)
4
>>> 'a' in x
False
>>> x[2][1:3]
'bc'
>>> x.count('a')
0
>>> x[2].split('c')
['ab', 'de']
>>> x.extend(100)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable

>>> x.extend([100])
>>> x
['abc', 'abcd', 'abcde', 'abcdef', 100]
```

2 The Function ShowCertainLength

Take a look at the module ShowCertainLength.py. Complete the implementation of the function CertainLength.

```
def CertainLength(L,m):
    """ Returns the number of strings in L that have length m.

    PreC: L is a list of strings and m is a positive int
    """
```



```
def CertainLength(L,m):
    """ Returns the number of strings in L that have length m.

    PreC: L is a list of strings and m is a positive int
    """
    k=0
    for x in L:
        if len(x)==m:
           k+=1
    return k
```

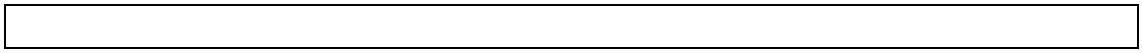
What is the output when ShowCertainLength.py is run?

```
len(L) = 109582

 m  ShowCertainLength(L,m)
----------------------------
 5         6919
 6        11492
 7        16882
 8        19461


len(L) = 5000

 m  ShowCertainLength(L,m)
----------------------------
 5          310
 6          461
 7          659
 8          825
```

3 The Function `TheLongest`

Take a look at the module `ShowTheLongest.py`. Complete the implementation of the function `TheLongest`.

```
def TheLongest(L):
    """ Returns the length of the longest string in L.
    PreC: L is a list of strings.
    """
```



```
def TheLongest(L):
    """ Returns a string from L that has maximal length.
    PreC: L is a list of strings.
    """
    lenMax = 0
    for s in L:
        if len(s)>lenMax:
            lenMax = len(s)
            sMax = s
    return sMax
```

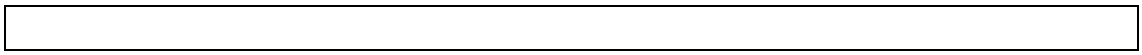
What is the output when `ShowTheLongest.py` is run?

```
len(L) = 109582

The longest string in L is antidisestablishmentarianism

len(L) = 5000

The longest string in L is antidisestablishmentarianism
```

4 The Function `SimplePlural`

Take a look at the module `ShowSimplePlural.py`. Complete the implementation of the function `SimplePlural`.

```
def SimplePlural(L):
    """ Returns a list of simple plurals

    An entry in L is a "simple plural" if it is the concatenation
    of the previous entry in L with the character 's'.

    PreC: L is a sorted list of strings each made up of lower case letters.
    """
```



```
def SimplePlural(L):
    """ Returns a list of simple plurals

    An entry in L is a "simple plural" if it is the concatenation
    of the previous entry in L with the character 's'.

    PreC: L is a list of strings each made up of lower case letters
    """
    T = []
    n = len(L)
    for k in range(1,n):
        s1 = L[k-1]        # The previous entry in L
        s2 = L[k]          # The "current" entry in L
        m = len(s2)
        if s2[m-1]=='s' and s1==s2[:m-1]:
            T.append(s2)
    return T
```

What is the output when `ShowSimplePlural` is run?

```
len(L) = 109582

The number of simple plurals is 15544


len(L) = 5000

The number of simple plurals is 774
```