CS 1110, LAB 8: FOR-LOOPS WITH LISTS, POSSIBLY NESTED

http://www.cs.cornell.edu/courses/cs1110/2017sp/labs/lab08.pdf

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

Getting Credit: Deadline: **in the first 10 minutes of (your) lab next week (Tue Mar 28 or Wed Mar 29)**. The checking-off procedure is the same as before.[1]

Create a new directory on your hard drive for this lab's files. Then, download into that new directory the files you need for lab 08; get them packaged in a single zip file from the Labs section of the course web page, http://www.cs.cornell.edu/courses/cs1110/2017sp/labs .

1. Writing for-loops for flat lists

Read the function specifications in lab08for.py, and look at the test cases for each such function given in lab08for_test.py to make sure you understand what each function is supposed to do. We've encoded the test cases in dictionaries to make them more readable and to reduce the tedium of typing them in.[2]

Then, finish the implementations of the functions in lab08for.py. You must make effective use of a for-loop in each one. Use any hints provided in comments. If you'd like some helpful for-loop examples, see lecture 11 and section 10.7 of the textbook. (Although that section is called "Map, filter, and reduce", it shows explicit for-loop patterns.)

You may test your code using lab08for_test.py. If the test cases there don't look complete, feel free to add your own.[3]

2. Writing for-loops for nested lists

Read the function specifications in lab08nested.py, and look at the two sample input lists given in lab08nested_test.py: `authors1` is sample input for `lab08.print_fam_name`, and `authors2` is sample input for `lab08.print_fam_andwork`.

Then, finish the implementations of the functions in lab08nested.py. You can run lab08nested_test.py to get a quick-and-dirty visual check that your code seems to be working correctly.

Course authors: E. Andersen, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

[1] In case you've forgotten, here's a reminder: Show this handout and/or your code to a staff member either (a) during your lab 08 session, (b) in consulting hours listed at http://www.cs.cornell.edu/courses/cs1110/2017sp/about/staff.php up to the day **before** your next scheduled lab section, or (c) in the first 10 minutes of (your) next scheduled lab (Tue Mar 28 or Wed Mar 29). Beyond that time, the staff have been instructed not to give you credit. Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

[2] Much as we love Cornell, typing "cornelltest" over and over again grows wearisome after a while. Loops are precisely the way programmers avoid needless repetition.

[3] Do note that the order in which the test cases are run will probably not the be the same as the order you see in the code. That's one aspect of for-looping through dictionaries that may take some getting used to.

3. For Reference: The Specifications Of The Functions To Implement

3.1. In lab08for.py.

```
def lesser_than(thelist,value):
    """Returns: number of elements in thelist that are strictly less than value.
    (Does not change thelist itself.)
    Preconditions: thelist is a list of ints, val is an int.

    Examples:  lesser_than([5, 9, 1, 7], 6) -> 2
    """
```

```
def uniques(thelist):
    """Returns: The number of unique elements in the list.

    Examples: unique([5, 9, 5, 7]) -> 3
              unique([5, 5, 1, 'a', 5, 'a']) -> 3

    Parameter thelist: the list to check (WHICH SHOULD NOT BE MODIFIED)
    Precondition: thelist is a list."""
```

```
def clamp(thelist,min,max):
    """Modifies the list so that every element is between min and max.
    (Does not return anything.)

    Any number in the list less than min is replaced with min.  Any number
    in the list greater than max is replaced with max. Any number between
    min and max is left unchanged.

    Example: if thelist is [-1, 1, 3, 5], then clamp(thelist,0,4) changes
    thelist to have [0,1,3,4] as its contents.

    Preconditions:
        thelist: possibly empty list of numbers (float or int)
        min, max: numbers, with min <= max"""
```

3.2. **In lab08nested.py.**

```
def print_fam_name(inlist):
    """Given list inlist whose items are sublists of corresponding to years,
    where each sublist is a list of two-item lists consisting of the
    first name and family name of an author born in that year,
    print all the family names (including repeats), one per line.

    Does not return anything, and does not alter inlist.

    Example input: look at the  list authors1 in lab08tested_test.py,
```

```
def print_fam_and_work(year_Author_list):
    """Given the possibly empty list year_Author_list of sublists,
    where each sublist is a list of Authors born in a given year,
    print the last_name of each Author contained in year_auth_list and, in
      the same line, the first of the works listed for that author.

    Does not return anything, and does not alter year_auth_list

    Example: for the list authors2 in lab08tested_test, one of the printout lines
       should be
            Steinbeck: Of Mice and Men
    """
```

The Author class is defined as having attributes `first_name`, `fam_name`, and `works`; see the docstring for the Author class for more detail.