CS 1110, LAB 08: FOR-LOOPS WITH LISTS, POSSIBLY NESTED

http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/lab08/lab08.pdf

First Name: ____________ **Last Name**: ____________ **NetID**: ____________

Getting Credit: **As always, strive to finish during the lab session** — it's the best way to stay on track in this course.[1]

Lab materials 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 from http://www.cs.cornell.edu/courses/cs1110/2018sp/labs .

1. Function lesser_than()

Your first task involves the function lesser_than(). For now, you just need to know that it takes two arguments and returns an int.

1.1. Understanding for-loops on dictionaries: encoding test cases.

We have given you test cases for lesser_than(), encoding them in a dictionary that you can find in function test_lesser_than() in lab08for_test.py.

Just as for a list thelist we can write for item in thelist:, we can loop through the keys of a dictionary.[2] Try this in Python interactive mode:[3]

```
d =  {"Lillian Lee": "LJL2", "Walker White": "WMW2"}
for key in d:
    print("The name is: " + key)
    print("The NetID is: " + d[key])
```

What do you get?

Lab authors: D. Gries, L. Lee, S. Marschner, W. White

[1]But if you don't manage finish during lab, here are the alternate checkoff opportunities: (a) at ACCEL Green room consulting hours, listed at http://www.cs.cornell.edu/courses/cs1110/2018sp/about/staff.php , **from today until Tue Mar 27 inclusive**, (b) at non-professorial TA office hours **from today to Wed Mar 28 3:45pm inclusive**, although at TA office hours, questions about course material or assignments take precedence over lab check-offs; or (c) during the **first 10 minutes of your next scheduled lab (Tue Mar 27 or Wed Mar 28)**. 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]More information is in Section 11.3 of the text.

[3]Hit the "tab" key to indent, and you will need to hit "return" twice after typing in the last line to convince Python you're done.

Here is the first few lines of the dictionary of test cases for lesser_than().

```
    test_cases = {((5, 9, 1, 7), -1): 0,
                  ((5, 9, 1, 7), 1): 0,
                  ((5, 9, 1, 7), 5): 1,  # etc.
```

The idea is that each *key* in this dictionary (the stuff before the colon in each line) represents a possible input, and each *value* (the stuff after the colon in each line) is the corresponding desired output.

Since `lesser_than` requires two inputs, a list of ints and an int, the keys are 2-item tuples. A technical condition of dictionaries is that lists can't be used as keys, so instead of the list [5, 9, 1, 7], we use the tuple (5, 9, 1, 7).

Using a dictionary makes it easy to add a testcase: you just have to add an extra line to the dictionary! And, the code for testing function lesser_than() has the following simple conceptual structure, or pseudocode:

```
for test_input in test_cases:
    decompose test_input into input_list and input_value
    result = lesser_than(input_list, input_value)
    test that result equals the expected answer, test_cases[test_input]
```

You can see how this is actually done in test function test_lesser_than() in lab08for_test.py.

1.2. **Check the test cases for lesser_than().** Here is the specification for lesser_than():

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

    Example:  lesser_than([5, 9, 1, 7], 9) -> 3
```

To check your understanding of that specification, we guarantee that exactly one of the test cases in the test-case dictionary, which starts around line 38 of file lab08for_test.py, *has the wrong desired output*. Which one is it, and what should the correct output be?

Remove (or correct) the incorrect test-case line in the dictionary before proceeding.

1.3. **Implement and test lesser_than() in file lab08for.py.** You can run lab08for_test.py to test your implementation.

If you'd like some helpful for-loop examples, see lecture 11 and its associated files and section 10.7 of the textbook — although that section is called "Map, filter, and reduce", it shows explicit for-loop patterns.

For your edification, we've provided two alternate implementations of lesser_than() in the lab files: they demonstrate how you can use filter() instead of an explicit for-loop to solve this problem.

2. Implement and test clamp() in file lab08for.py

You can see test_clamp() in lab08for_test.py to see some test cases, and run the file to test your implementation.

(GO TO NEXT PAGE)

3. Function perfects() in file lab08for.py

The specification of our next function is:

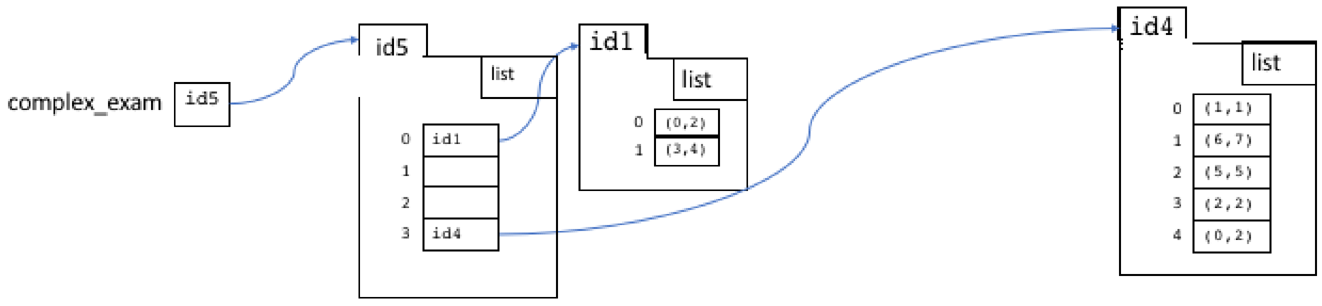
```
def perfects(exam):
    """Returns: the number of perfect subquestions answered on this `exam`.

    A graded exam consists of a non-empty list of questions.
    Each question is a non-empty list of 2-tuples of the form
         (got, poss)
    where `got` is the int score that the student received, and `poss` is the
    int max score achievable on that problem, 0 <= got <= poss.
```

3.1. Understanding the input nested lists. The code in test function test_perfects() in file lab08for_test.py constructs the following test instance, consisting of four questions:

[[(0, 2), (3, 4)], [(0, 2), (4, 4)], [(3, 3), (7, 7), (5, 5)], [(1, 1), (6, 7), (5, 5), (2, 2), (0, 2)]]

and stores (the identifier of) this list in variable complex_exam. It has 7 perfect scores.

We started diagramming the above list for you. **Add the missing objects and values.**



3.2. implement and test perfects(). You can run lab08for_test.py to test your implementation.

4. Strictly optional question

You don't need to do this part of the lab to get checked off! But if you would like extra practice, try implementing uniques() in lab08for.py. To test it, uncomment the line that appends test_uniques to the list of functions to test in lab08for_test.py (you can search for the word "uncomment" to find the exact location).