Homework 5: Complexity and HOF

Total Points: 100 (Correctness and Style)+ 3 EC (Checkpoint)

Due Dates:

- **Entire Assignment: Tuesday, February 13th, 11:59pm**
- **Checkpoint (read below): Sunday, February 11th, 11:59pm**

Starter Files

Download hw05.zip. Inside the archive, you will find starter files for the questions of this homework. You cannot import anything to solve the problems.

IMPORTANT: Coding and Docstring Style

This is a reminder that your code style will be graded. Here are a few useful links:

Style Guide Document

Style Guide on Course Website

Style Guide Examples

Testing

At any point of the homework, use one of the following command to test your work:

```
>>> python3 -m doctest hw05.py
>>> py -m doctest hw05.py
>>> python -m doctest hw05.py
```

Checkpoint Submission

Due Date: Sunday, February 11th, 11:59pm (SD time)

You can earn up to **3 points extra credit** by submitting the checkpoint by the due date above. In the checkpoint submission, you should complete **Question 1, 2** and submit the hw05.py file to gradescope.

Checkpoint submission is graded by completion, which means you can get full points if your code can pass some simple sanity check (no tests against edge

cases). Note that in your final submission, you should still submit these questions, and you may modify your implementation if you noticed any errors.

General Notes and Requirements

1. **DO NOT IMPORT ANY PACKAGES.**
2. Please add your own doctests (**at least three**) as the given doctests are not sufficient. **You will be graded on the doctests**.
3. When a question requires **assert statements**: follow the assert guide to write assert statements to prevent any unexpected inputs. You will be graded on this. If a question does not require assert statements, assume all inputs are valid.

Required Questions

As the weekend approaches, you and your friends are heading to the bigO island, a new-found island in Pythonia, and embarking on a mission to unravel the mysteries of time complexity!

Question 1:

Complexity is an important concept in coding as we want our code to be as efficient as possible. Judge if the following equations about complexity hold. Write your answer (`True` or `False`) in the *complexity_tf()* function, which returns your answers as a list of 10 boolean values. In this list, write your answer to the first equation at index 0, second equation at index 1, etc.

Equation Format: f(x) = O(g(x))

(Note for students who have this question: **We use big-O as a tight bound** in DSC20, not upper bound).

Examples:

1. $3n = O(n)$ True
2. $\log_2 n = O(1)$ False

Note: The doctests for this question are sanity checks, only indicating that you have put answers in the correct form.

Equations:

1. $\sqrt{\frac{n \cdot (n-2)}{3}} = O(n)$
2. $n \cdot \log n + (log(n))^3 + n^2 = O(n \log n)$
3. $\log_2(\log_2 n) + \log_5 n = O(\log_4 n)$
4. $n \cdot log^2(n) + n \cdot log(n) = O(n \log n)$
5. $\frac{15^n}{14^n} = O(1)$
6. $\frac{n^2(6n^2+10)}{2n} = O(n^2)$
7. $1 + 2 + 3 + 4 + \ldots + n/4 = O(n)$
8. $\frac{n!}{(n-2)!} = O(n)$
9. $\log_5 n^n + 15n = O(n \log n)$
10. $n^5 + 5^n = O(5^n)$

```
def complexity_tf():
    """
    Write your answers to time complexity True/False questions in this
    Function. No new doctests required.

    >>> answers = complexity_tf()
    >>> isinstance(answers, list)
    True
    >>> len(answers)
    15
    >>> all([isinstance(ans, bool) for ans in answers])
    True
    """
```

```
    # REPLACE ... WITH YOUR ANSWERS (True/False) #
    return [...]
```

Question 2:

Continuing with the concept of complexity, recall the order of common time complexities:

1. O(1) - constant time
2. O(log n) - logarithmic time
3. $O(\log n)^k$, k>1 - Poly-log time
4. O(n) - linear time
5. O(n log n) - linearithmic time
6. $O(n^2)$ - quadratic time
7. $O(n^3)$, $O(n^4)$, etc. - polynomial time
8. $O(a^n)$, a > 1 - exponential time
9. O(n!) - factorial time
10. None of the above

Read the source code of 15 python functions, and judge the time complexity of them. Write your answer (as the order 1 - 10, duplicates allowed, each number may or may not be used) in the *complexity_mc()* function, which returns your answers as a list of 10 integers. In this list, write your answer to the first function at index 0, second function at index 1, etc.

Example:

```
def example(n):
    s = 0
    for i in range(5, n+1):
        s = s + 1
```

Explanation (simplified but the complexity is still the same) (you do not have to write it but make sure you understand your choice):

- There is 1 instruction that assigns a 0 to a variable s.
- Also there is a single loop that runs *n–4* times.
    - Each time the loop runs it executes 1 instruction in the loop header and 1 instruction in the body of the loop.
- The total number of instructions is $1 + 2 * (n - 4) + 1$ (for the last loop check) $= 2n - 6 = O(n)$

For more examples: click here

Please note: Java syntax is used in their loops. You can read it as:

Java	Python
for (int i = 0; i < n; i=i+1) • *i* is initialized to a 0. • if *i* is less than *n*, perform one iteration of the loop • increment *i* by 1, check if *i* is less than *n*. ○ If it is, run the loop again etc; ○ If *i* is greater than *n*, stop the loop.	*for i in range(0, n)* Equivalent loop in Python

Notes:

1. For each function, if its parameter is ***n***, you can assume it's a non-negative and **super large** integer; if its parameter is a list (*lst*), you can assume ***n*** is the length of this list, also **super large**.
2. The doctests for this question are sanity checks, only indicating that you have put answers of the correct form.

```
def complexity_mc():
    """
    Write your answers to time complexity MC questions in this
    function. No new doctests required.

    >>> answers = complexity_mc()
    >>> isinstance(answers, list)
    True
    >>> len(answers)
    15
      >>> all([isinstance(ans, int) and 1 <= ans <= 10 for ans in
answers])
    True
    """
    # REPLACE ... WITH YOUR ANSWERS (1-10, duplicates allowed) #
    return [...]
```

Question 3:

You finished all questions about complexity, so it is time to sail back from the big-O island. On way back, a captain gave you an encoded message so that you have something to do on a way back.

Write a function that takes three unique characters `key1, key2, key3` (as length-one strings) and returns an inner function that takes in a string `cipher_text` and returns a decoded new string (all in lowercase).

How to decipher: Once you see a letter that is the same as one of the input characters (*case insensitive*, meaning B is the same as b), convert it with following rules:

Rule One:

- The letter is key1 and
- The index of the character is odd -> convert to letter 'i'
- The index of the character is even -> convert to letter 'e'

Rule Two:

- The letter is key2 and
- The index of the character is odd -> convert to letter 'o'
- The index of the character is even -> convert to letter 'a'

Rule Three:

- The letter is key3 -> convert to letter 'u'

Example:

- `Key1 is Q`
- `Key2 is b`
- `Key3 is C`
- Original string is bNsWqR

Input	Return	Explanation
b	a	key2 (b), index value is even (0) -> a (output)
N	n	Not one of the three keys, stays the same (but lowercase) -> n
s	s	Not one of the three keys, stays the same (but lowercase) -> s
w	w	Not one of the three keys, stays the same (but lowercase) -> w
q	e	key1 (q), index value is even (4) -> e (output)
R	r	Not one of the three keys, stays the same (but lowercase) -> r

```
def decipher_text(key1, key2, key3):
    """
```

```
>>> decipher_text('Q', 'b', 'C')('bNswqR')
'answer'
>>> decipher_text('W', 'J', 'Z')('TESTYJZRCJDE')
'testyourcode'
>>> decipher_text('W', 'J', 'Z')('ONWTWST')
'onetest'
>>> decipher_text('D', 's', 'c')('')
''
"""
```

Question 4:

To help students understand how they are doing in DSC20, Marina wants to create a function that could return what student's current grade for their lab and homework based on the information provided to that function.

Write a higher-order function that takes in two float numbers (weights) between 0 and 1 representing how much the lab and homework are worth (sum to 100%). And returns **two** functions, both functions take two tuples of integers representing the score you got for your labs and homeworks.

- First function would return your current grade as a numeric number between 0 and 1, round to 2 decimal places (use `round()`).
- Second function would return it as a letter grade, where grade >= 90% is A, 90% > grade >= 80% is B, 80 > grade >= 70 C, 70 > grade >= 60 D, otherwise F.

The grade would be calculated as below, where:

- N_1 is the number of labs you finished,
- x_n is your grade for the nth lab, as a number between 0 - 1.
- N_2 is the number of homeworks you finished
- y_n is your grade for the nth homework, as a number between 0 - 1
- And w_1, w_2 is the percentage of how much your lab/homework is worth.

$$grade = \frac{w_1}{N_1}\sum_{n=1}^{N_1} x_n + \frac{w_2}{N_2}\sum_{n=1}^{N_2} y_n$$

Notes:

1. All numbers in the parameter would be between 0 and 1 (inclusive)

2. If labs/homework is an empty tuple, assume that you get 0 for your lab/homework

Example:

```
# lab would worth 40% and homework worth 60%
grade, letter_grade = grade_calculator(0.4, 0.6)

# all your labs got 100% and your homeworks got 50%, 100% and 0%
# final grade would be 0.4 * average of labs + 0.6 * average of
homeworks
# 0.4 * (1 + 1 + 1) / 3 + 0.6 * (0.5 + 1 + 0) / 3 = 0.7
grade((1, 1, 1), (0.5, 1, 0))

# calculate the letter grade for 70% numeric grade, which is ‘C’
letter_grade((1, 1, 1), (0.5, 1, 0))
```

```
def grade_calculator(w1, w2):
    """
    >>> output = grade_calculator(0.4, \
    0.6)
    >>> len(output) == 2
    True
    >>> output[0]((1, 1, 0.5), (1, 1, 1))
    0.93
    >>> output[1]((0.6, 0.7, 0.8), (0.8, 0.8, 0.8))
    'C'
    >>> output[1]((0.5,), (1,))
    'B'
    >>> output[1]((), ())
    'F'
    >>> output[0]((), ())
    0.0
    >>> output[0]((1,), (1,))
    1.0
    >>> output[0]((1,), ())
    0.4
    """
```

Submission:

By the end of this lab, you should have submitted the lab via Gradescope. You may submit more than once before the deadline; only the final submission will be graded.