Object-Oriented Programming
50:198:113 (Spring 2019)

Homework:	**3**	**Professor:**	**Suneeta Ramaswami**
Due Date:	**3/10/19**	**E-mail:**	`suneeta.ramaswami@rutgers.edu`
Office:	**323 BSB**	**URL:**	`http://crab.rutgers.edu/~rsuneeta`
		Phone:	**(856)-225-6439**

Homework Assignment 3

The assignment is due by 11:55PM of the due date. The point value is indicated in square braces next to each problem. Each solution must be the student's own work. Assistance should only be sought or accepted from the course instructor. Any violation of this rule will be dealt with harshly.

This assignment requires you to write some recursive functions and implement your own `Time` class. As usual, you are graded not only on the correctness of the code, but also on clarity and readability. Hence, I will deduct points for not following the guidelines for your class design, poor indentation, poor choice of object names, and lack of documentation. Every module, class, and method should be properly documented. For other code, use a common sense approach, i.e., *do not* document every line of code and only include a *brief and clear* explanation if you think the code may not be self-explanatory. **Please read the submission guidelines at the end of this document before you start your work.**

Problem 1 [30 points] Recursive functions. In this problem, you are asked to write three recursive functions. All three functions should be implemented in a module called `problem1.py`. In each of these functions, you **may not** use any built-in functions other than `len`, the index operator `[]` , the slice operator `[:]`, and `+` (string or list concatenation). You also may not use any loops. *Please note:* The built-in function `in` is essentially a loop and may not be used in any of the functions in this question.

1. [10 points] Write a recursive function called `replace_element` with three parameters: `L` (a list), `oldel`, and `newel`. The function returns a list in which every occurrence of `oldel` in `L` has been replaced with `newel`. For example, `replace_element([5, 4, 3, 5, 1], 5, 100)` should return the list `[100, 4, 3, 100, 1]`. Keep in mind that all restrictions stated above apply for the recursive implementation.
2. [10 points] Write a recursive function called `num_double_letters` with a single string parameter `astr` that returns the number of occurrences of double letters in `astr`. A double letter is simply a consecutive pair of the same character. For example, `num_double_letters("mississippi")` should return 3 because double letters occur 3 times in the string (`"ss"`, `"ss"`, and `"pp"`). Note that if the same letter occurs consecutively three or more times (this doesn't happen too often in English!), we count every double letter pair. For example, `num_double_letters("hmmm")` should return 2 because there are two double `m` pairs (the second and third letter, as well as the third and fourth letter), and `num_double_letters("hmmmm")` or `num_double_letters("mmhmmm")` should return 3.
3. [10 points] Write a recursive function called `has_repeats` with a single list parameter `L`, that returns True if `L` has repeating elements (i.e., an element occurring more than once) and False otherwise. For example, `has_repeats([3, 2, 1, 5, 12])` should return `False` and `has_repeats([3, 5, 2, 1, 5, 12])` should return `True`.

Problem 2 [20 points] Time class. In this problem, you are asked to design a class for `Time` objects. Note that Python has a `datetime` module that contains a `datetime` class for time and date objects. In this problem, you are asked to implement your own `Time` class to represent objects that are times of day. You will download two files from the Sakai site for this problem: `problem2.py` and `test_time.py`. We will implement the following methods for this class.

1. `__init__`: The constructor initializes the hour, the minute, and a string indicating whether it's an AM or PM time. **The constructor has been implemented for you in the file `problem2.py`.** Please look at the instance attributes specified in the constructor before implementing the remaining methods. Note that the constructor ensures that only valid `Time` instances are allowed.
2. `hour`: Returns the hour of the `Time` instance.
3. `minute`: Returns the minute of the `Time` instance.
4. `am_pm`: Returns a string (`"AM"` or `"PM"`) indicating whether the `Time` instance is an AM or PM time.
5. `total_minutes`: Returns the total number of minutes from midnight until the `Time` instance. For example, if `T` is the `Time` instance 12:30PM, then `T.total_minutes()` should return 750.
6. `__str__`: Returns a printable version of the `Time` instance, which is a string representation of that instance. Keep in mind that the printable version of a `Time` instance must always use two digits for the minute. For example, if `T = Time(9, 7, "am")`, then `print(T)` should print `9:07AM`.
7. `__repr__`: Returns a meaningful representation of the `Time` instance. This should be the string representation in the form described for the `__str__` method

As stated previously, complete the implementation of the `Time` class in the file `problem2.py` (*you must download this file from Sakai*). I have also provided a test file called `test_time.py` that you can use to test your implementation. That module will simply import the `problem2.py` module. Edit `problem2.py` to implement the `Time` methods described above. When you are ready to test your implementation, type `python3 test_time.py` to test your implementation.

Submission Guidelines

Please name the module for each problem as specified in the problem description above. In particular, create a module called `problem1.py` for Problem 1, and `problem2.py` for Problem 2 (**download this module from Sakai**). Also, *please make sure that your name and RUID appear as a comment at the very top of the file.*

Submit your homework files via Sakai as follows:

1. Use your web browser to go to the website `https:sakai.rutgers.edu`.
2. Log in by using your Rutgers login id and password, and click on the `OBJECT-ORIENTED PROG S19` tab.
3. Click on the 'Assignments' link on the left and go to 'Homework Assignment #3' to find the homework file (`hw3.pdf`), and the modules `problem2.py` and `test_time.py` (for Problem 2).

4. Use this same link to upload your homework files (`problem1.py` and `problem2.py`) when you are ready to submit.

You must submit your assignment at or before 11:55PM on March 10, 2019.