Homework 6: Dictionary Order vs. Lexicographic Order

(Due Date and Points are as specified in Ctools)

Your goal in this assignment is to use conditional statements to provide a function that compares two strings in dictionary order (or phone book order). The reason for writing this function is that both Python and C++, by default, compare strings in lexicographic order, based on ASCII values. Thus, as an example, in Python, the following holds:

"XYZ" < "abc"

Above violates dictionary order. The reason above is true in Python is because 'X' has a smaller ASCII value than the ASCII value of 'a'. Lexicographic comparison simply compares corresponding characters from both strings from left-to-right based on ASCII values. Whenever the characters are different, the answer is computed based on the comparison at that position. For example:

"abc" < "abde" (because 'c' < 'd')

"abd" > "abcdefgh" (because 'd' > 'c')

Lexicographic order gives the same order as in a dictionary if the two strings are both of the same case (i.e., both lower case or both upper case). However, it fails to give the correct ordering if the cases are not identical. So, one strategy for comparing two strings in dictionary order is to:

- Convert both strings to the same case (either both lower or both upper)
- Then compare the converted strings lexicographically. If this gives an ordering, then accept that ordering as the answer. In the first example of comparing "XYZ" and "abc", we would end up comparing "xyz" with "abc" after conversion to lower case. This would give the ordering that "abc" is smaller than "xyz" lexicographically. From that, we can conclude that "abc" comes before "XYZ" in dictionary order.
-
- If the two strings turn out to be equal after being converted to the same case, then we need to be careful. For example, if we compare "xyz" with "XYZ", then the two strings will be equal after converting to lower case. In this instance, we want "XYZ" to be smaller than "xyz" because 'X' is considered to be smaller than 'x'. In a Phone book, typically, entries that start with capital letters come before those with entries that start with lower letters.

What you are given:

You are given a file dictorder.py.It contains the following functions:

- compare_dictionary_order(s1, s2). This function is supposed to return:
  - -1 if s1 comes before s2 in a dictionary (phone book)
  - 0 if s1 and s2 are identical strings

- +1 if s1 comes after s2 in a dictionary (phone book)

You will be revising the code for this function. This is what is graded.

- test_compare_dictionary_order(): This is simply a sample test function. You can add additional tests if you wish. But, we are not grading this function.
- main() function: This simply calls test_compare_dictionary_order(). Python programs don't require a main() function. But, it is a good practice to include one for code readability.
- Call to main() at the end of the file. This causes main() to run, which calls test_compare_dictionary_order(), which then calls compare_dictionary_order.

What you need to do

1. Change the first function, compare_dictionary_order(s1, s2) in dictorder.py to meet the above specification and as described in the comments inside the function.

   Test the function and make sure it passes all the tests. To test the function, you can simply do:

   ```
   % python  dictorder.py
   ```

2. Once your program works from step 1, you can get an estimate of your score on the assignment. Simply run:

   ```
   % python grade.pyc
   ```

   Note: grade.pyc has the pyc suffix. These are binary version of python files and not readable.

3. Make sure that your code in the function compare_dictionary_order(s1, s2) is commented well and uses good coding style. Simplify the code and write good comments to get full style points (See the lecture on Conditional Statements on possible simplification strategies).
4. Submit the modified dictorder.py. Also run the program with grade.pyc and take a screenshot that shows your auto-graded score out of 8 points.

This assignment should not be a long one. The function can be written within 10-60 minutes once you understand the specifications and have written a recipe using pencil and paper. If it takes you much longer in translating a recipe to a program to Python code, you should probably seek advice in office hours/lab/lecture or do pair programming with another student in the course.

If you have trouble understanding the specifications or coming up with a recipe in English or pseudo-code, even with the explanation above and the explanation in the comments, then also seek some help. Once you understand the specs and the recipe, the coding part should be relatively quick.