Project 1: Alphabetization

Due: Friday Jan 25, 11:59 pm

1 Assignment Overview

You are the manager for your club's email roster. It is your responsibility to keep it ordered so that people can easily find particular members. Unfortunately, not everyone knows everyone's full name. You must thus create two lists; one sorted by first name and one by last name.

You know that once you allow this functionality the other e-board members will have other orders that they want you to complete. You have decided that you will create only one method that uses a supplied ordering to sort the members.

Additionally, club membership has grown tremendously in recent years and the club president is worried about how long it will take to make these lists. She wants you to both generate the sorted list as quickly as possible and to quantize the amount of effort required to create it.

2 Assignment Deliverables

You must turn in completed versions of the following files:

- alphabetizer.py

In addition, for procedural reasons you need to upload the following files (but don't need to modify them):

- main.py
- gryffindor.txt
- sorted_first_name.txt
- sorted_last_name.txt

Be sure to use the specified file name and to submit your files for grading via **Mimir** before the project deadline.

3 Assignment Specifications

Your task will be to complete the methods listed below:

- `order_first_name`
- `order_last_name`
- `is_alphabetized`
- `alphabetize`

Your implementation of `is_alphabetized` should run in $O(n)$ time and your implementation of `alphabetize` should run in $O(n \log n)$ time. There is an obvious solution that can be done using a $O(n^2)$ algorithm. Though it may be a good approach for initial problem solving, you are *required* to implement a faster solution with $O(n \log n)$ times for full credit. To demonstrate this, your `alphabetize` method will return the number of comparisons made in addition to the alphabetized roster.

Additionally, your alphabetization should be stable. This means that if two people have the exact same name (both first and last) then they should have the same order in the resulting list as they do in the original list.

You should include comments where appropriate. In particular, you **must** identify how you alphabetize the membership roster.

4 Assignment Notes

- Points will be deducted if your solution has warnings of any type.
- You have been supplied with stubs for the required methods. You must complete these methods to complete the project. You may add more functions than what was provided, but **you may not modify the signatures of the provided methods**.
- You have also been supplied with functions that read and write the member list from a file as well as a main function for executing your code.
- You do not have to worry about error checking for valid input. You may assume that the supplied reader function correctly reads the input file.
- Remember that not all names are unique. For example, siblings will often have the same last name.
- The `ordering` parameter is a function pointer. You can invoke it with `ordering(a, b)` just like you would for a regular function.
- You may not use the `sort`, `sorted`, or similar functions.
- Your code should follow the Python style guide. Test 0 will check that you are following this style.
- Please submit the whole directory. Some of the unit tests will read the given input files.
- 5 points will be assigned manually. In particular, points will be given for documention. You must document how you are alphabetizing the roster.