CSCI 4041, Fall 2018, Programming Assignment 2

Due Tuesday, 9/18/18, 10:30 AM (submission link on Canvas)

This is not a collaborative assignment; you must design, implement and test the solution(s) on your own. You may not consult or discuss the solution with anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class. Obtaining or sharing solutions to any programming assignment for this class is considered academic misconduct. If you are not sure what this means, consult the class syllabus or discuss it with the course instructor.

You are placed in charge of the email servers as part of your unpaid internship at Holistic Synergies, Ltd. Your supervisor, Dr. Boss Manager III, has compromised company security by leaking data to hackers over email on multiple occasions. However, he has a solution. After watching a certain sci-fi movie series, Dr. Manager has come to the conclusion that hackers always use short, trendy names like “Neo” or “Cypher”. Therefore, rather than displaying emails in order from newest to oldest, he orders you to reprogram the server to display emails from longest sender name to shortest, so that all of the emails from people with short hacker names are pushed to the bottom.

Speed is of the utmost importance to Holistic Synergies, Ltd., so you will be implementing this email sorting routine in Quicksort. However, Dr. Manager believes that his ancient enemy, Ted from Accounting, will take advantage of Quicksort’s $O(n^2)$ worst-case runtime to slow down his email by sending many consecutive emails from aliases of ever-increasing length. So, he also requires that a backup implementation be created using Merge Sort.

Download the template PA2.py from the class website. The template includes a `Email` class, which consists of two instance variables: `sender`, which is a string representing the name of the person or entity who sent the email, and `subject`, which is a string representing the subject line of the email. The file also includes some test cases representing sample inboxes for Holistic Synergies, Ltd. employees. You’ll need to implement Merge Sort and Quicksort algorithms which operate on a list of Email objects, and sort them in non-increasing order by the length of the `sender` string.

Requirements:

- You must download the template file PA2.py and edit the `merge`, `merge_sort`, `partition`, and `quick_sort` functions. Do not edit any other part of the file.
- Your program must run without errors on the version of Python installed on the CSELabs machines, Python 3.5.2. (if you’re testing this on CSELabs, you need to type python3 or idle3 instead of python or idle to start the correct version from the terminal)
- You are not permitted to use any built-in Python sorting routines like the `sorted()` function or the `.sort()` list method. You are also not allowed to use any Python function that asks for user input, such as `input()`, as this will break the grading script.

You may not import any modules other than traceback, which is already imported by the template.

- You must implement the Quicksort and Merge Sort algorithms as described in the textbook. Any other sorting algorithms will receive very little credit, even if you pass every test case.
- However, note that while the textbook algorithms describe how to sort a list of numbers in non-decreasing order, this problem requires you to sort a list of Email objects in non-increasing order by length of sender name, so you will need to adjust the algorithm slightly.
- The textbook version of Merge Sort is a stable algorithm; your Merge Sort must also be stable; the final Merge Sort test case checks this. Quicksort is not stable, so yours does not have to be.
- This assignment will be graded automatically based on the number of test cases your program passes. There will be several secret test cases in addition to the ones included in the template to ensure you're not hard-coding in the solutions.
- The grading breakdown for this assignment is as follows:
    - 30%: File runs without syntax errors
    - 70%: Passing test cases without breaking any requirements.
- The unedited template file already runs without syntax errors and passes 6/12 test cases (because 6 of the 12 test cases are lists that are already in sorted order). This means that if your program causes syntax errors or passes less than 6/12 test cases, you will get a better score by just submitting the original template unedited.
- Submit your edited PA2.py file to the Programming Assignment 2 link on Canvas before 10:30 AM on 9/18/18. No credit will be given for late submissions.