Programming Design, Spring 2016
Homework 12

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

Please upload one PDF file for Problem 1 and two CPP files for Problems 2 and 3 (optional) to PDOGS at http://pdogs.ntu.im/judge/. Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***2:00 am, May 23, 2016***. Please answer in either English or Chinese. The maximum point of this homework is 120.

Before you start, please read Chapters 8 and 18 of the textbook.[1]

The TA who generates the testing data and grades this homework is ***Wei-Hung Liao***.

Problem 1

(20 points; 5 points each) Please answer the following questions regarding the class `string`.

(a) Use your own word to describe how to use the member functions `find`, `find_first_of`, and `find_first_not_of`. Explain how they are different from each other.

(b) Make your own implementations of operator overloading for `==`, `!=`, `<`, `<=`, `>`, and `>=` by using the member function `compare`.

(c) Make your own implementation of the member function `clear` by using the member function `erase`.

(d) Implement a member function

```
size_t getSpaceCount(const size_t from, const size_t to) const;
```

that returns the number of white spaces in between the character positions `from` and `to` (including them) in a string. Explain why there are three `const`, how they are different, and why there is no `const` at the beginning of the function header (as part of the return value).

Problem 2

(55 points) Continue from Homework 11, let's add more functions into the car sharing information system.

Now when one rates the other, she may also leave a review with at most 500 characters. These characters are restricted to be capital English letters, lowercase English letters, white spaces, or punctuation marks including commas, periods, exclamation marks, question marks, and semicolons. In particular, there is no newline character in a review.

Using these reviews, the company wants to penalize drivers/passengers who are late with a high probability. For a driver/passenger, let her *probability of being late* be the percentage of received reviews containing the word "late", capital or lowercase (so "LATE", "laTE", "LaTe", etc., all count). Note that if one does not receive a review after a ride, it is considered as receiving a review with no content. Therefore, when we calculate the probability of being late, the numerator is the number of reviews containing the word "late", capital or lowercase, and the denominator is the total number of rides involving this user. Note also that we are only asking whether the string "late" exists in the review, capital or lowercase. It does not matter whether it is a word or part of a word. For example, if the review is "I will use the service later.", it also counts as a review containing "late".[2] The two reviews

[1] The textbook is *C++ How to Program: Late Objects Version* by Deitel and Deitel, seventh edition.
[2] If you think this is unreasonable, you will fix it in Problem 3.

are given at the end of a ride, i.e., in event A.

The the probability of being late, the assignment rules are modified as follows:

- Passenger: In Homework 11, the rule is that if a passenger's *average rating* is strictly lower than 2, the system will disallow her to request for a car. Now the rule is still the same, except that the satisfaction indicator becomes

  $$\text{the average rating} - 5 \times \text{the probability of being late.}$$

  We call this quantity her *score*. For example, if one's average rating is 4.2 and her probability of being late is 10%, her score is $4.2 - 5 \times 0.1 = 3.7$.

- Driver: In Homework 11, the rule is that a driver is assigned to a passenger according drivers' distance to the passenger and their average ratings. Now we use their scores instead of average ratings, where a driver's score is calculated in the same way as a passenger's score.

You will be given a sequence of events. By processing these events, you will update the statuses of the passengers and drivers. At the end you should print out the IDs of all searching drivers, in the order of their registration times.

Input/output formats

There are 15 input files. In each file, there are $n + 2m + 1$ lines, where $n \leq 10000$ is the number of events and $m \leq 10000$ is the number of type-A evetns. Except the input for event A, all others are in the same format as in Problem 2 of Homework 11. After a line of event A, there are two lines of characters representing the reviews left FOR the driver and passenger, respectively.

For example, consider the following input:

```
4
P 1
P 2
D 1
D 4
D 3
O 1 0 0
S 1 10 10
S 2 5 5
A 1 300 300 5 2
This driver is awesome!
He was LATE!!!!
C 1
O 1 10 11
O 4 10 10
O 3 6 6
S 2 10 9
S 1 10 10
D 2
O 2 -1 -1
C 3
```

Let's process these events:

- First two passengers and three drivers sign up. Then driver 1 opens the app at $(0, 0)$. When passenger 1 then searches for a driver, driver 1 is assigned. As no other drivers are available, when passenger 2 then searches for a driver, he will not be served and will give up. Passenger 1 (and driver 1) then arrives $(300, 300)$, driver 1 gets a rating 5, and passenger 1 gets a rating 1. Driver 1 then closes the app and later reopens it at $(10, 11)$.

- Then drivers 4 and 3 open the apps at $(10, 10)$ and $(6, 6)$. When passenger 2 later searches for a driver at $(10, 9)$, all the three drivers are available. First we calculate the three drivers' distances to the passenger, which are 2, 1, and 7. As $M = 4$, the system will pick one driver between drivers 1 and 4. As driver 1's average rating (5) is greater than driver 4's (3, by default), driver 1 will be assigned, even though driver 4 is closer to the passenger.
- Now passenger 1 searches for a driver again. Unfortunately, his score is $2 - 5 \times 1 = -3$ is strictly lower than 2. Therefore, his order is rejected, and he gives up.
- Driver 2 signs up and opens the app at $(-1, -1)$, and driver 3 closes the app.

As we may see, at the end only drivers 4 and 2 are available. Therefore, the output should be

```
4 2
```

In general, the IDs of available drivers should be printed out according to their registration times, where two consecutive values should be separated by a white space. As driver 4 signs up earlier than driver 2, the output should be `4 2`, not `2 4`.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use only techniques covered so far. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

You must use the given classes to store the given input information. If you fail to do so, you will get no point. If you do, you will be graded according to the following rule:

- 45 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 3 points.
- 10 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.

Problem 3

(45 points) This problem is a complication of Problem 2. Now a review is considered as containing "late" only if the string "late" exists in the review and it is a complete word. In other words, the characters before and after it must be either a white space or a punctuation mark. All other requirements are the same.

Input/output formats

Exactly the same as those in Problem 2. For example, for the input

```
4
P 1
P 2
D 1
D 4
D 3
```

```
O 1 0 0
S 1 10 10
S 2 5 5
A 1 300 300 5 2
This driver is awesome!
He was LATEEEEEEE!!!!
C 1
O 1 10 11
O 4 10 10
O 3 6 6
S 2 10 9
S 1 10 10
D 2
O 2 -1 -1
C 3
```

Passenger 1 is not considered as receiving a review containing "late". Therefore, when she searches for a car, she will get assigned driver 4. It then follows that the output of this problem is

```
2
```

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use any technique. You are not even required to use the classes defined in Problem 2. Finally, you should write relevant comments for your codes.

Grading criteria

45 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 3 points.