CSE201 Advanced Programming (CSE Section)
Lab Assignment 01

IIIT-Delhi. 9th August 2018. Due by 11:59pm on 10th Aug 2018

This is a take-home lab assignment. No extensions whatsoever will be provided. Any submission after the deadline will not be evaluated. If you see any ambiguity or inconsistency in a question, please seek a clarification from the teaching staf. Please read the entire text below very carefully before starting your implementation.

Plagiarism: All submitted lab assignments are expected to be the result of your **individual** effort. You should never misrepresent someone else's work as your own. In case any plagiarism case is detected, it will be dealt as per IIITD plagiarism policy and without **any** relaxations: https://www.iiitd.ac.in/sites/default/files/docs/education/AcademicDishonesty.pdf

Please note that you are **not** allowed to discuss the design/solution of the lab assignment (e.g., backpack discussions, etc.). Anyone who is found doing this will be treated as a plagiarism case.

Topics Covered: Lectures 01 and 02

You have been hired to design a client assignment/management system for a civil law firm. Each lawyer in the firm specializes in one of the four different kinds of legal disputes handled by the firm, namely - **family law, defamation, housing and finance**. A lawyer only takes up clients whose case belongs to his/her field of specialization. The words client/case will be used interchangeably henceforth, as each client is associated with exactly one case. The following rules must be followed while deciding which lawyer should be assigned a given client:

1. The client's case will be taken by the law firm for consideration only if the category of legal dispute belongs to any one of the four mentioned above.
2. The case can only be assigned to a lawyer specializing in the same category of legal dispute. **Assume that for each kind of legal dispute, there is at least one lawyer specializing in it. The case will be assigned to the lawyer having the least number of pending cases.**
3. Each lawyer will have his/her own automated client manager who will show the next case to handle on the basis of following two priority rules:
    a. Each client will be assigned a priority level beforehand by the people at the front desk. This priority can be determined from a lot of factors like date of hearing, type of client, etc. You, as the system designer, do not need to worry about these factors and simply accept a user-provided priority level. There are five levels of client priority in this firm (1-5, in decreasing order of priority).
    b. **Clients belonging to the same priority level will be processed on first-come-first-serve basis**. (Assume that a lawyer in this firm can only work on one case at a time.)

4. After a client has been assigned a priority level, no one else should be able to change the priority level. It must remain fixed till the end.

Following are the different kinds of queries which must be handled by the system:

0. *Register a new lawyer*
All lawyers will be registered before any other query is made. Inputs: lawyer name, specialization. Each lawyer should be assigned an unique id.

1. *Register a new client*
   This query will involve assigning an unique id, a priority level and a lawyer to the client. Inputs: client name, category of legal dispute, and priority level. All client names will be single words. Display the unique id, and name of lawyer assigned.
2. *Show next case to handle for a given lawyer*
   This case should be decided on the basis of the priority rules mentioned earlier. This operation should remove this case from the list of pending cases for that lawyer. Input: lawyer id. Display the name of the client, category of legal dispute and his/her unique id.
3. *Show details of the current client being handled by a given lawyer*
   Input: lawyer id. Output the same details of the client as query 2.
4. *Show all pending cases of a given lawyer*
   Input: lawyer id. Output the same details of the client as query 2 for each pending case/client.
5. *Show details of all lawyers in the firm*
   Display the name, specialization and number of pending cases of each lawyer.
6. *Remove a client*
   Input: client id.

Note: The system should not crash for any kind of invalid input. It should handle it properly, and display an appropriate error message. However, the following input format will always be followed. You should strictly follow object-oriented programming approach in your implementation as discussed in class lectures. There is no need to show us the sequence diagram.

Input Format:
The first line will have an integer *n* for the number of lawyers in the firm.
Each of the next *n* lines will have two space separated strings: lawyer name and specialization.
The next line will have an integer *q*, the number of queries.
Each of the next *q* lines will have variable number of space-separated inputs. The first input in the line will always be an integer specifying the query type. The rest of the inputs in the line will depend on the type of query to be performed.

Sample Case:

Input:

```
4
lawyer0 family
lawyer1 defamation
lawyer2 housing
lawyer3 finance
9
1 client0 housing 3
1 client1 housing 5
1 client2 housing 3
1 client3 finance 4
4 30
2 30
6 345
2 40
3 10
```

Output:

```
10
20
30
40
12345 lawyer2
996 lawyer2
12 lawyer2
345 lawyer3
client0 housing 12345
client1 housing 996
client2 housing 12
client0 housing 12345
No pending cases! // Error message
No current case! // Error message
```