Middle East Technical University | Department of Computer Engineering

CENG 140

C Programming

Fall' 2020-2021
Take-Home Exam 3

Emre Külah
kulah@ceng.metu.edu.tr
Due date: Jan 27, 2021, Wednesday, 23:59

MASTER CENG

1 Overview

Our department decided to organize a challenge called MasterCeng. Students will develop 5 different projects and submit their codes in a single submission. A jury formed from our instructors will evaluate each project and give them a score. Each submission will be evaluated as sum of its projects scores. Average score of the challenge and the winner will be announced.

The jury will score the projects according to programming language used to develop the project. The instructors have weights for each programming languages:

Instructor	**C**	**C++**	**Python**	**Java**
Sengor	2.4	2.6	4.2	4.0
Ahmet	4.4	4.3	2.2	3.0
⋮	⋮	⋮	⋮	⋮
Tolga	3.8	5.0	1.2	1.3
Instructor-n	3.3	3.7	2.9	4.5

Jury will get the submission, the programming languages that projects written in, and will score each project according to programming languages.

	Project-1	**Project-2**	**Project-3**	**Project-4**	**Project-5**
Submission	C	Python	Python	Java	C++

Before evaluating the submission, the jury members have to be decided. Let's say size of the jury is 3 and Instructor-1, Instructor-2 and Instructor-n are selected.

```
Jury-1 Weights (Instructor-1) => 2.4 2.6 4.2 4.0
Jury-2 Weights (Instructor-2) => 4.4 4.3 2.2 3.0
Jury-3 Weights (Instructor-n) => 3.3 3.7 2.9 4.5
```

So calculation will be:

```
Project-1: Jury-1 Weight + Jury-2 Weight + Jury-3 Weight => (2.4+4.4+3.3) (written in C)
Project-2: Jury-1 Weight + Jury-2 Weight + Jury-3 Weight => (4.2+2.2+2.9) (written in Python)
Project-3: Jury-1 Weight + Jury-2 Weight + Jury-3 Weight => (4.2+2.2+2.9) (written in Python)
Project-4: Jury-1 Weight + Jury-2 Weight + Jury-3 Weight => (4.0+3.0+4.5) (written in Java)
Project-5: Jury-1 Weight + Jury-2 Weight + Jury-3 Weight => (2.6+4.3+3.7) (written in C++)
```

So final score for the submission will be:

```
(2.4+4.4+3.3) + (4.2+2.2+2.9) + (4.2+2.2+2.9) + (4.0+3.0+4.5) + (2.6+4.3+3.7) = 50.8
```

Different jury members will be selected for each submission's evaluation. At the end, average score for the challenge and the winner will be announced.

2 Structs

- Instructors and submissions will be stored in **linked lists**.
- Projects of submissions will be stored as an **array of structure** in each submission.
- Scores of submissions will be an **array of structure** as well.

```
struct Instructor
{
    float *pl_ratings;
    char *instructor_name;
    struct Instructor *next;
};
typedef struct Instructor Instructor;

struct Project
{
    int pl_index;
    char project_index;
};
typedef struct Project Project;

struct Submission
{
    Project *projects;
    char *student_name;
    struct Submission *next;
};
typedef struct Submission Submission;

struct Score
{
    char *student_name;
    float score;
};
typedef struct Score Score;
```

3 Tasks

- **void insert_instructor(Instructor **list, char *name, float *pl_ratings):** function gets instructors list pointer, name of the instructor and PL ratings of the instructor as parameter. It is going to add new instructor into the list of instructors with their programming language ratings.
- **void insert_submission(Submission **list, char *name, int *pl_list):** function gets submissions list pointer, name of the student and a list of integers of size 5, containing programming language of corresponding project. It is going to add new submission into the list of submissions with corresponding list of projects.

  Programming languages are enumerated as follows:

  ```
  C: 0
  C++: 1
  Python: 2
  Java: 3
  ```

- **Score* calculate_submission_scores(Instructor* instructions, Submission *submissions, int** juries, int jury_size):** function takes instructors, submissions, jury members as **2D integer array** and size of jury members as parameter. For each submission, **jury_size** number of jury members will rate the submission.

  It is going to calculate scores of the submissions and going to save them into a dynamically created array of **Score** structure. Vertical size of the juries array is not provided since it is equal to size of submissions' linked list.

  2D juries array will be in following format:

  Let's say we have 4 submissions and jury_size is 3.

  ```
  [
      [jury_index1_in_instructors, jury_index2, jury_index3],  \\ for submission 0
      [jury_index1_in_instructors, jury_index2, jury_index3],  \\ for submission 1
      [jury_index1_in_instructors, jury_index2, jury_index3],  \\ for submission 2
      [jury_index1_in_instructors, jury_index2, jury_index3],  \\ for submission 3
  ]
  ```

- **float find_average_score(Score *scores, Submission *submissions):** function takes scores and submissions as argument. It is going to calculate average score for challenge, which is the average score computed overall submissions.
- **Score find_winner(Score *scores, Submission *submissions):** function takes submissions and scores as argument. It is going to find winner with the highest score and it is going to return corresponding **Score** instance.
- **void print_instructor(Instructor *instructors):** function takes instructors' linked list and prints in the following format:

```
Name: #name_of_the_instructor_1#
Ratings: #c_rating#, #c++_rating#, #python_rating#, #java_rating#
Name: #name_of_the_instructor_2#
Ratings: #c_rating#, #c++_rating#, #python_rating#, #java_rating#
...
Name: #name_of_the_instructor_n#
Ratings: #c_rating#, #c++_rating#, #python_rating#, #java_rating#
```

- **void print_submission(Submission *submissions):** function takes submissions' linked list and prints in the following format (Project details have a whitespace at the beginning):

```
Name: #name_of_the_student_1#
 Project-1: #programming_language_of_project1#
 Project-2: #programming_language_of_project2#
 Project-3: #programming_language_of_project3#
 Project-4: #programming_language_of_project4#
 Project-5: #programming_language_of_project5#
Name: #name_of_the_student_2#
 Project-1: #programming_language_of_project1#
 Project-2: #programming_language_of_project2#
 Project-3: #programming_language_of_project3#
 Project-4: #programming_language_of_project4#
 Project-5: #programming_language_of_project5#
```

- **void print_scores(Submission *submissions, Score *scores):** function takes array of Score structure and prints each item in the following format:

```
Score of #name_of_the_student_1#: #score_of_submission_1#
Score of #name_of_the_student_2#: #score_of_submission_2#
...
Score of #name_of_the_student_n#: #score_of_submission_n#
```

4 Example

INSTRUCTORS				
	C	C++	Python	Java
Sengor	5.0	4.2	2.3	3.1
Hande	2.1	3.4	4.2	5.0
Ahmet	3.1	5.0	4.5	2.3
Tolga	3.2	4.2	1.3	4.6

Figure 1: Instructors

STUDENTS & SUBMISSIONS					
	Project-1	Project-2	Project-3	Project-4	Project-5
Emre	C	C++	C	Python	C++
Gorkem	C++	Python	Java	C	C
Yusuf	Python	Python	Java	Java	C++

Figure 2: Submissions

JURY MEMBERS (SIZE: 3)			
	Jury-1	Jury-2	Jury-3
Emre	Sengor	Hande	Ahmet
Gorkem	Sengor	Ahmet	Tolga
Yusuf	Hande	Ahmet	Tolga

Figure 3: Juries

```
Output of print_instructor
-----------------------------------------------------
Name: Sengor
Ratings: 5.00 4.20 2.30 3.10
Name: Hande
Ratings: 2.10 3.40 4.20 5.00
Name: Ahmet
Ratings: 3.10 5.00 4.50 2.30
Name: Tolga
Ratings: 3.20 4.20 1.30 4.60

Output of print_submission
-----------------------------------------------------
Name: Emre
 Project-1: C
 Project-2: C++
 Project-3: C
 Project-4: Python
 Project-5: C++
Name: Gorkem
 Project-1: C++
 Project-2: Python
 Project-3: Java
 Project-4: C
 Project-5: C
Name: Yusuf
 Project-1: Python
 Project-2: Python
 Project-3: Java
 Project-4: Java
 Project-5: C++

Output of scores
-----------------------------------------------------
Score of Emre: 56.60
Score of Gorkem: 54.10
Score of Yusuf: 56.40

-----------------------------------------------------
Average score: 55.70
Best student: Emre, Max Score: 56.60
```

5 Specifications

- If two or more of the submissions get same score, the winner of the challenge will be the one with the minimum index.
- Floating points will be printed with 2 digit precision.

6 Regulations

- **Programming Language:** C
- **Libraries and Language Elements:**
  You should not use any library other than *"stdio.h"*, *"stdlib.h"*. You can use conditional clauses (switch/if/else if/else), loops (for/while), allocation methods (malloc, calloc, realloc). **You can NOT use any further elements beyond that (this is for students who repeat the course).** You can define your own helper functions.
- **Compiling and running:**
  DO NOT FORGET! YOU WILL USE ANSI-C STANDARTS. You should be able to compile your codes and run your program with given **Makefile**:

```
>_ make the3
>_ ./the3
```

- **Submission:**
  You will use CengClass system for the homework just like Lab Exams. You can use the system as an editor or work locally and upload the source files. Late submission IS NOT allowed, it is not possible to extend the deadline and **please do not ask for any deadline extensions**.
- **Evaluation:** Your codes will be evaluated based on several input files including, but not limited to the test cases given to you as an example. You can check your grade with sample test cases via CengClass system but do not forget it is not your final grade. Your output must give the exact output of the expected outputs. It is your responsibility to check the correctness of the output with the invisible characters. Otherwise, you can not get a grade from that case. If your program gives correct outputs for all cases, you will get 100 points.
- **Cheating: We have zero-tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations and will get 0. Sharing code between each other or using third party code is strictly forbidden. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.