CS 409/509 Advanced C++ Programming - Project

Due: 10 June 2019 @ 23:55

Teaching Assistant Scheduling Framework

Description:

Your project is to create a "teaching assistant scheduling framework" in C++17 standard.
You can either use Clang or GCC compiler in Linux environment. MSVC is not accepted.

You will program an application on top of your framework that will read data from two files that are in the same folder with the application: COURSES.csv and ASSISTANTS.csv. Application will then write the solution to a SOLUTION.csv file.

COURSES.csv:

```
CourseCode1,InstructorName,MinTAHours,MaxTAHours,MinTACount\n
CourseCode2,InstructorName,MinTAHours,MaxTAHours,MinTACount\n
…
```

For Instance:

```
CS101, Kubra Kalkan Cakmakci, 60, 100, 3
CS321, Furkan Kirac, 10, 20, 1
CS409, Furkan Kirac, 10, 20, 2
```

ASSISTANTS.csv: this file will contain the name of the assistant and his/her required hours to assist per week. HoursToAssists can either be 5, 10, 15, 20, or 30.

```
NameOfAssistant1, HoursToAssist, MaxCoursesToAssist, OldAssistedCourses, …\n
NameOfAssistant2, HoursToAssist, MaxCoursesToAssist, OldAssistedCourses, …\n
…
```

For Instance:

```
Ahmet, 20, 3              // 20h/week req., at most 3 courses, no old experience
Aytekin, 10, 2, CS101     // 10h/week req., at most 2 courses, assisted CS101 once
Ebru, 20, 3, CS321        // assisted CS321 once
Baris, 30, 3, CS321, CS409, CS409, CS409  // assisted CS321 once and CS409 for 3 times
Zeynep, 30, 3             // …
Hasan, 10, 2
```

Application must write its solution to a file in the same folder whose name is SOLUTION.csv.

SOLUTION.csv:

```
CourseCode1-SectionCode, NameOfAssistant, HoursToAssist, NameOfAssistant, HoursToAssist, …\n
CourseCode2-SectionCode, NameOfAssistant, HoursToAssist, NameOfAssistant, HoursToAssist, …\n
…
NameOfAssistantX, HoursFreeX,
NameOfAssistantX, HoursFreeY,
…
```

For Instance:

```
CS101, Ahmet, 20, Ebru, 10, Hasan, 10, Zeynep, 20
CS321, Baris, 20
CS409, Ebru, 10, Aytekin, 10
Baris, 10
Zeynep, 10
```

Note that Zeynep's 20 hours out of 30 is assigned to CS101. An assistant can be partially assigned to multiple courses in valid partitions of 5, 10, 15, 20, or 30 h/week. But he/she can only be assigned to the *MaxCoursesToAssist* number of courses. Baris and Zeynep each have 10 hours/week unassigned time slots. It is written to the end of the solution file.

You will be graded out of 100 points. Listed below are your enhancements to be coded and their respective grade points.

Grading requirements	**Points**
Using Modern C++ coding guidelines for reading, storing and supplying data to solver classes: Almost always auto, class definitions, Rules of Big Four (copy-swap idiom), templates, curiously recurring template pattern (CRTP) usage in a meaningful place, etc. For instance, define classes for Course, Assistant, Data, Solver, etc. Data can be CRTP defined that can access a later defined Solver. You may choose any design you wish.	30
Implement a greedy heuristic solution to the problem which is not optimal. **Create a GreedySolver class** for this. For instance, start from the first assistant and keep assigning his/her weekly hours to the next course to be assisted.	20
Define a cost function "C = penalty – bonus" as below: Assistant with old course experience assigned to a new course -> 5 pts penalty per unexperienced course Assistant assigned to 2 courses at the same time -> 5 pts penalty Assistant assigned to 3 or more courses at the same time -> 20 pts penalty Course requirement not met -> 100 pts penalty For each X h/week unassigned assistant time -> X pts bonus **Find the best solution that minimizes this cost function**. **Create OptimalSolver class** for this. If you cannot find the real best you will get zero grade from this section. Therefore, a heuristic method might not work with the test data we will plug into the application. You might want to try a clever exhaustive search method.	20
Consider that we may plug real data. Number of courses and number of assistants may be around 20 and 50 respectively. In a conventional Intel core-i7 laptop with quad cores and 16GB RAM: Your application completes in at most 1 second. (30 pts) Your application completes in at most 3 seconds. (20 pts) Your application completes in at most 10 seconds. (10 pts) Else (0 pts)	30

Submission Information:

Send all your source codes to the LMS. Your code should be clean and easy to read by possessing the following properties;

- *Clean structure:* The overall code should be neatly organized, where the related statements are grouped together with enough spacing among them.
- *Appropriate use of comments:* There should be comments explaining what the program, and different groups of statements are supposed to do. Don't overdo it.
- *Meaningful and consistent variable naming:* The names of variables should be meaningful with respect to the purpose and usage of these variables.

Submission: By uploading your code and report to LMS as a single ZIP archive. No other methods (e.g., by e-mail) accepted. (You may resubmit as many times as you want until the deadline).
Warning: DO NOT SHARE YOUR CODE WITH OTHERS. Your programs are checked and compared against each other using automated tools. Any act of cheating will be punished severely. The code that does not compile will receive 0 points.
Also:

- Name your archive file uploaded exactly as requested. Your archive file must be named as **<NAME>_<SURNAME>_<STUDENTID>.zip.**
- Make sure that your program runs and gives the **expected output.**
- The first lines of your code must include your name, surname, student number, and department as a **comment**. An example comment is as follows:
  / John Smith S0001 Department of Computer Science */*

Good luck ☺