IN2029 Programming in C++
Coursework

There is a single coursework in this module, counting for 30% of the overall module mark. This coursework provides practice with classes and the containers, iterators and algorithms of the C++ Standard Library. It does not require any features covered after session 6.

This coursework is due at **5pm on Sunday 5rd December**. As with all modules, this deadline is hard. (See "Submission" below for the procedure for applying for extensions.)

Overview

You are to implement a scoring system for a particular sport. Your program will receive reports of new scores for players, and then rank them according to various measures.

I have prescribed a particular external structure and names for your classes: please follow these precisely, because I will be testing your solutions automatically. The private parts of your classes, and implementations of functions, and any other functions you may choose to add, are up to you.

Description

You should implement and test the following classes.

`class record`

with constructors

- **`record()`**
  set up an empty record.
- **`record(double score)`**
  set up a record with one score.

and the following public methods:

- **`void add_score(double score)`**
  record a new score for the player. Scores are guaranteed to be non-negative.
- **`double best_score()const`**
  returns the best score ever added to the record (or 0 if none)

- `double overall_average() const`
  returns the average of all scores added to the record (or 0 if none)

- `double recent_average() const`
  returns the average of the last 10 scores added to the record (or 0 if none). If there are fewer than 10 scores in a record, it returns the average of those scores.

- `bool novice() const`
  return whether fewer than 10 scores have been recorded.

You should try to avoid holding more values than necessary. (The **deque** container may be helpful.)

`class table`

with a default constructor and methods

- `void add_score(const string &name, double score)`
  adds a new score for the named player.

- `int num_players() const`
  returns the total number of players for whom a score has been recorded.

- `vector<string> best_recent() const`
  returns the names of the players in descending order of recent averages.

- `double average_best() const`
  returns the average of the best scores of all players. You may assume that at least one player has been added.

- `string best_overall() const`
  returns the name of the player with the highest overall average. You may assume that at least one player has been added.

- `int novice_count() const`
  returns the number of novice players. Your implementation should use a library algorithm.

Marking

The classes will be worth 40% each.

The remaining 20% of the marks will be for clean programming style, including consistent layout, sensible identifier names, useful comments, avoidance of superfluous variables and data members, and general clarity of code.

Submission

Submit a ZIP file containing your source files only, to Moodle.

If you believe that you have extenuating circumstances that justify an extension, you should

- submit your work to `smcse-extension@city.ac.uk` within one week of the deadline (i.e. by 5pm Sunday 12th December), including your student number, course, module code and the original due date, **and**
- apply for an extension through the Extenuating Circumstances process.

As not all claims for extenuating circumstances meet the criteria, you can also submit on Moodle what you have by the deadline, as a fallback in case your claim is rejected.