Assignment Concurrency 22/23 (draft)

Reza Hassanpour
Ahmad Omar
Afshin Amighi
Andrea Minuto

UPDATE: Deadline and delivery link.

Subject:

The course of concurrency requires creating a concurrent effective **Exam Running System** that runs for several students and teachers.

Groups:

An assignment can be carried out by a group of 1 or 2 students (no exceptions).
All group members must be within the same class (same teacher).
Each group needs to provide the implementation of a **concurrent** program that can prepare and execute an “exam with a list of questions” (see the scenario).

Scenario:

There is a *classroom* with a list of *students* and *teachers*. An *exam* is assigned for the *teachers* to design, and the same *exam* is assigned to the *students* to perform. Teachers start designing and adding *questions* to the exam (or exams) in the first step. When the *exam* is ready, it will be assigned to the *students*. *Students* will pick each *question* and propose their *answers*. When all the *students* finish their *exams*, statistical information about the *exam* will be reported.
There is only **one** classroom with a list of students (default: 200), all the students are in the classroom, and each participates in one exam. The exam is made by all the teachers (default: 10), and each teacher proposes the maximum amount of questions (default: 40). For each question, every student must propose **one** answer.
Note: changing parameters will alter this example accordingly.
By this example, the exam consists of 400 questions, each teacher will produce 40 questions, and there will be 200 students giving 400 answers (one per question all in the same shared exam) each.

Assignment details:

Exercise: Run the sequential version and check the execution time. There is a need for a more efficient/concurrent program **without changing the algorithm executed (meaning: changing it will be an instant fail, be it removal of sleep functions, printout or any other alteration of the algorithm)** by teachers and students.
Be aware that not all the classes will need to be implemented concurrently. Recognising which classes needs to be parallelised is part of your assessment.

Your assignment will be to add concurrent features to the provided package using the following guidelines:
- There will be **multiple teachers and multiple students**.

- The number of teachers, students and questions are provided as **fixed** parameters. Feel free to change them for your experiments, but **they must have their original values in the final submission**.
- To test the increase in efficiency, you must employ **several concurrent teachers and students**.
- The shared resource needs **to support simultaneous access safely.**
- Given code contains to-do lists/comments. Follow them and implement your solution.
- **Overprotection of variables and creating unnecessary threads will result in failure** (overprotection: protecting areas of memory that do not require protection)**.**
- Programming Language: C# is the official language for the project.
- Your submission ***must contain values for SubmissionParams***. *Check the provided code.*
- *There is no need to add interactivity (such as: asking for parameters at runtime) with the user.*
- *All the assignments will be tested with the command "dotnet run" from the command line in the project root directory.*

What will be evaluated:

- Respecting the requirements above.
- Minimal decency in the code quality.
  - Error at runtime as "null pointer exceptions" or any other runtime error from unhandled data, rewritten memory or other is not considered a sufficient assignment even if it is an easy fix.
- The program should be thread-safe.
- The printed result (**total number of answered questions**) for the concurrent version must **match** the **sequential** one.
- It should have a proper concurrent implementation (not serial). The assignment will receive a passing grade if the concurrent implementation reflects all the course teachings:
    - it must implement a **thread-safe concurrent program**.
- The delivery modality must be respected.
- You are not allowed to use parallel-for, task library (and similar libs), async-await or other message-passing ready-made libraries.

Submissions:

Instructions for the submissions:

- Submission Deadline: **29/01/2023 23:00**.
- The submitted file MUST contain the group's requested information. Check class *SubmissionParams* for more details.
- The delivery should be in a zip file named: *studentnumber1_studentnumber2_infcoc.zip* (the file name saved online will add the name of the submitting account).
- The delivery zip should be *renamed* from ".zip" to ".dot". Ex: *studentnumber1_studentnumber2_infcoc.zip —> studentnumber1_studentnumber2_infcoc****.dot***
- ***In the delivery, there should be no trace of any runtime compiled file. Solutions should be cleaned (bin/obj should be removed)***
- Submission Procedure: upload the files as instructed in the form at the following link.
    - https://forms.office.com/e/kJfjm2KjYc

Reflection points:

- **Parameters**:
  There are two classes defined for parameters.
  - The values of the *SubmissionParams* must be determined by the student(s) in the final submission.
  - The values of the *FixedParameters* must not change in the final submission. However, students can change the values of FixedParams for their experiments. For example, lowering *maxNumOfStudents* can give a quick result. But, the **final submission must keep the original values**.
- The implementation requires different tuning depending on your processor.
  A lot of testing is required.
- Altering the original data in the delivery, and adding external resources of any kind **is forbidden**. Variation of the existing parameters is encouraged but should be put back in the original values once done.
- Be aware that the delivery is final. You can deliver anytime before the deadline. **Resubmission is not possible** (consider it an exam in class).