Assignment Concurrency 22/23 (retake)

Reza Hassanpour
Ahmad Omar
Afshin Amighi
Andrea Minuto

Subject:

The course of concurrency requires creating a concurrent effective **Packaging System** that runs for workers who are asked to pack a collection of items within boxes.

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 a “concurrent packaging system” (see the scenario).

Scenario:

There is a *storage* with a list of *items* and *boxes*. In the sequential version, a worker is responsible for picking up an item labelled with its corresponding box, moving it to the place where boxes are set up, and packing it within the corresponding box. In the end, each box must have only items labelled with the box name, the number of items picked must be equal to the number of items packed and all the items within the boxes must be equal to the original number of items.
There is only **one** storage with a list of items, a list of boxes and one worker.
Items are assigned randomly to the boxes.

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

Assignment details:

Your assignment will be to add concurrent features to the provided package using the following guidelines:

- There will be **multiple items and multiple boxes**.
- Packaging with one single worker is time-consuming. There will be **multiple concurrent workers** to improve **the packaging time**.
- The number of boxes, items and workers 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 workers**.
  - To facilitate the testability of the code, there is a function (called: “*EvaluateConc()*”) that allows you to conduct tests in batches. It is up to you to make correct use of the provided tools.
- The shared resources need **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 *(.Net 6.x)*.
- 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 (.Net 6.x).*

What will be evaluated:

- Your concurrent solution must improve the performance of the sequential version.
- Your concurrent solution must have correct values for the final statistical parameters. Check provided method: *EvaluateConc(…).*
- Your concurrent solution must have correct thread creation, join, and shared resource protection.
- 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 printed result (**total number of items, boxes, etc**) for the concurrent version must **match** the **sequential** one.
- 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), concurrent queue, concurrent lists, async-await or other message-passing ready-made libraries.

Submissions:

Instructions for the submissions:

- Submission Deadline: **Friday, June 02, 11:30 pm**.
- 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***
- ***There should be no trace of any runtime compiled file in the delivery. 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/RzEYPBdFyj

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 *maxNumOfItems* 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.
- The original sequential version should not be altered.
- Be aware that the delivery is final. You can deliver anytime before the deadline. **Resubmission is not possible** (consider it an exam in class).