FINAL:
The link and deadlines have been updated.

Retake assignment

Pickup ~~*restaurant*~~ Library!

The setting:

We are setting up a **fast** pickup Library! The books are so good that they can't stay in the library for more than a few seconds! There are a huge number of books available, but they are all different. The customers go to wait at the library until the book they are waiting for is ready at the counter.
They read it at home and bring it back as soon as possible!

In this library, there are two locations where customers and clerks can exchange books read and read. The first one is where you go to the counter to pick up the book, and the second one is where you drop off your read book.

The clerks are behind the counter and furiously run up and down to load the books ready for the customers, and then wait for a delivery at the drop-off to put the book back in the record. Every clerk *only brings one book to the counter* (records its absence) and brings back *one single book to the registry for every run*. After they are done with one delivery to a customer and archive the one they can find at a drop-off (only one), they can go home and rest.

The code:

The provided code might or might not work. It is not implemented concurrently. Your duty as a developer is to complete correctly the given application. The application should become **multithreaded** reaching the maximum level of concurrency, so everything that can be done in a concurrent setting should be implemented that way.

This **concurrent** application (command line only) requires no interaction with the user. Any interaction with the user is **forbidden**.

You must grant the property of **fairness** and **correctness** in your assignment.

Every part of your code must be completed so no random interleaving can alter the outcome. So, it must be **thread-safe**.

Exceptions are indicated in code: do not alter any of the code marked as not to be touched, it does not matter if you think it is not ok, or if it should be safe as well and it is not. Consider it a legacy constraint. Part of the code that is marked as "do not alter" even if they are NOT thread safe, should remain as such.

If the strategies to protect your memory are excessive (read: overlocking, or locking things that should not or do not need locks of any kind), they will be evaluated as invalid. Try to remember that is important to minimise the coverage of your locking (or number of semaphores). *Spinning* is not allowed.

The desired outcome is:

- Each **clerk** brings one book (and only one) from the records to the counter location and marks it as borrowed, then takes a small sleep, and removes one and only one book from the drop-off location to the records to check it back in.

- Each **customer** will wait to get a book at the counter location, read it (take a small sleep), and bring it back to the drop-off location. After that, it will terminate. The customers will get one and only one book.

A precondition for the assignment is that the number of customers and clerks is always the same at the beginning. So each of them will just run once and never be reused. This is a simplification for handling the concurrent work. No checks for different situations should be evaluated.

There is no need to check for speed of execution. But take care that each task gets executed by who should do it (remember to check the flat scoping problem with the lambdas).

Breaks (“thread.sleep") are there for specific purposes and should not be removed or moved from their logical spot. YOU SHOULD NEVER ADD MORE. Similarly, printouts are there for testing and logical purposes. Do not remove them or alter them in the final delivery. During testing, feel free to debug it in the way you prefer, even altering or adding I/O operations, remember to clean up your code before the final delivery. Forgotten lines of code that should not be there will be counted as mistakes.

The main method will start elements in phases (already indicated in the method), feel free to change the content of the functions call as needed.

You can create new classes if needed.

You can add variables if needed.

You can alter the signatures of some of the methods (where allowed) if needed.

We highly encourage testing: rise and/or diminish the number of threads. Remember that testing over different memory conditions (over extra load/with less load) can facilitate the evidence of some errors or abnormal behaviours.

The provided code has comments to guide you and help you address where to edit and where not to edit. Try to add as less changes as needed. Respect the structure and be considerate of what the comments are saying.

***Note*:** the printout of the stats must always print correct values (no books on the counter, no books left in the drop-off, and the number of available books must be the same number of books in the record.

Note for who did the previous assignment: the code looks very similar, but it is NOT the same, do not try to “copy-paste” the solutions from before without thinking about the local situation first.

The delivery:

Every delivery group is made of **up to 2 students.** Only one student must submit the file containing the cleaned solution. It is not allowed to work across classes unless explicitly approved. If the classes are joined (shared theory and/or practice) it is possible to cooperate without explicit approval. Retakers can choose anyone, no approval is needed.

In Microsoft Forms, you will find the information on how to enter the data. The names on the code will represent who is part of the group. Your code MUST include the names and student numbers of each component.

The delivery will be in Microsoft Forms. Late submissions, email submissions, and chat submissions will be discarded.

Deliveries must NOT contain the **bin/obj or any other temporary file** or directory.

LINK to the delivery: https://forms.office.com/e/qhh8NYwFCJ

Deadline: **June 25, 2024 at 11:30 pm**.

Note on Microsoft forms:

Changes to the submission are possible in Microsoft Forms, due to technical issues, any change after the deadline *might* invalidate the submission. So, do not make changes to the submission after the deadline.

When asked, at the end of the submission form, SAVE the link in your forms, so you can edit the submission from there.

Only one submission (editable) per student is allowed.

FOLLOW THE INSTRUCTIONS ON THE SUBMISSION TO UPLOAD THE FILE.

The file ".zip" must be renamed to ".dot" to workaround the limitations of the form.

The testing and evaluation:

If your assignment fails to meet any of the following points, your grade might get a **negative assessment** with no further testing.

- You do NOT require user interaction.
- Your code compiles correctly and does NOT require versions of ".Net" other than ".Net 6.x".
- Your code does NOT show any syntax errors.
- Your code does NOT crash.
- You are NOT using an auto-locking data structure (concurrent queues, blocking data structures, other data structures, or primitives that handle threads automatically, ex.: task library or "promises-like" functions of different kinds).
- You are NOT using parallel-for or other facilitating in-language control systems that avoid using threads in a raw context. Async-await structures are part of this category and thus: NOT allowed.
- You are NOT using spinning to synchronise threads (avoid spinning).
- You are NOT using external libraries that are not the standard Threads/ mutex/locks/semaphores.
- Your code is cross-platform (the solution will NOT only run on your system).
- You understand every single part of your code and which are the implications (to keep in mind: according to the course description an oral check is possible at any time to clarify your contribution).
- Your code is original (to keep in mind: according to the course description an oral check is possible at any time to clarify your contribution).
- Your delivery does NOT contain *bin* and *obj* directories or other temporary files.
- Your code implements what is requested.
- Your code does NOT implement some kind of a sequential version of the task provided.
- The delivery format is as requested (file naming, compression method for the solution, etc…).

The usage of any code-generating facilitation is forbidden (**chatgpt**-like or other autogenerating systems). Keep in mind that platforms of that kind tend to give always similar results to each other.

Code linter/autocomplete—> ok

Copilot-like—>**forbidden**.

What will be tested?

Besides the previous requirements and minimal decency of code quality, we will check that **all the shared memory is correctly protected** and that the algorithm that synchronises the various tasks is carried out correctly by who should do it (and not altered). Your code must be granting fairness and correctness in the application (for fairness, it should be all delegated to the OS, which is assumed fair). We will check if the **protection** applied is sufficient and **only where necessary**, and evaluate negatively what is overprotected.

How will it be tested?

Partially automatically and partially by hand. The “automatic” testing will be on compiler errors/warnings, formatting of the files, and printout results (in case the reviewer needs to, will also run these tests by hand). The “by hand” part will be carried by command line calls as “dotnet run” in the project directory. Are you using Visual Studio? Consider testing if everything is working by running the code via the command line interface. So, you can be sure that everything is OK, and with the same setting as the provided project.

Common questions:

Do you doubt a concept relevant to your code? You can ask in the forum, but please, refrain from asking questions (in public or private) as “Is this ok?” or “Is it sufficient like this?”. Teachers are not allowed to answer such questions to prevent unfair evaluation.

Do you not know where to start? Check all the exercises, we are making use of the information needed there, and we used them and discussed them in class.

I forgot XYZ in the delivery, can I change it? You will have time to change it till the deadline, and past the deadline, the assignment will be considered as delivered. Make use of the link editing in the delivery, if you lose it we can’t generate it for you.

I forgot to fill in the name of my partner, can I add it? You will have time to change it till the deadline, and past the deadline, the assignment will be considered as delivered, with or without edits.