Universitetet i Oslo
Institutt for Informatikk
Research group for Reliable Systems (PSY)
Andrea Pferscher, Einar Broch Johnsen, Silvia Lizeth Tapia Tarifa, Riccardo Sieve

INF 5170: Models of Concurrency

Fall 2024

Mandatory assignment 1

23. Sep. 2025

Due: Monday 27. Oct. 2025 (23:59 Norwegian time)

General remarks

Language

You must use English.

Before delivery

You must *work in groups of exactly 2 people* (neither more nor less). The groups remain the same as in the first oblig.

How to deliver

- Your solution should be delivered online (https://devilry.ifi.uio.no/).
- All solutions, including program examples, must be commented in order to make them understandable for the group teacher.

Who delivers

For "technical" reasons (devilry): in a group each member should upload the same solution (which should be identical, just the same PDF uploaded several times). The solution must be marked with names and email addresses of all contributing students.

Check in time that devilry works and that your status within devilry (and student web etc) is OK. Do not try your IN5170-devilry access as late as the day of the deadline. In case of doubts, for clarifications, or if having trouble with devilry etc, ask in time.

Evaluation

This assignments are graded *pass* or *fail*. You must pass the obligs in order to take the final exam.

1 The Roller Coaster problem

Exercise 5.17 (a) in Andrews. A number of *passengers* (1:n) wish to repeatedly take rides with a *Roller Coaster*. There should be one process for each passengers and one for the Roller Coaster. As for the problem of the *sleeping barber*, the processes must participate in several synchronization stages. The passengers must wait for the car to be present, and the car must wait for `C` passengers to enter. After the ride, the car must awake all riding passengers. The solution does not have to prevent sneaking. Make a solution in the setting of *monitors with signal and continue discipline* for signaling.

Use the await language to solve this exercise.

In addition, try to formulate a reasonable invariant based on your solution, but it is *not* necessary to prove the invariant using programming logic.

2 "Thread-safe" queue as linked list with Go

Implement Task 3 of Oblig 1 in Go. More specifically, the task is to implement a simplified map-reduce system that computes the sum of all squares up to a certain integer, using a thread-safe linked queue and thread pools. The system has the following components:

- An input queue that has to be filled with the first n numbers.
- A queue for all even numbers.
- A queue for all odd numbers.
- Two *mappers*: each mapper takes one number from the input queue and adds its square to the even queue (if the number is even) or to the odd queue (if the number is odd).
- Two *reducers*: each reducer is assigned one queue (either odd or even) and computes the sum of all elements.
- One thread pool to fill the input queue
- One thread pool to take values from the input queue and distribute them to the two mappers
- One thread pool to take values from the even and odd queue and transmit them to the correct reducer (the even numbers to the "even reducer", the odd numbers to the "odd reducer").

You will find a skeleton of a Go application that realizes this architecture online[1]. Please fill in the marked places in the source code to realize the above behavior as necessary to ensure thread safety. You should pay attention to the following.

- Submit executable code.
- Synchronize where you have to do so.
- Don't synchronize where you don't have to! For example, the three tasks (inserting in the input queue, mapping and reducing) should be done concurrently.
- Both mappers should insert in the even as well as in the odd queue.

[1] `https://www.uio.no/studier/emner/matnat/ifi/IN5170/h25/obligs/obl2skel.go`