UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF4140 — Models of Concurrency
Day of examination:	15. December 2008
Examination hours:	14.30 – 17.30

This problem set consists of 5 pages.

Appendices:	None
Permitted aids:	All written and printed

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

Some general advises and remarks:

- This problem set consists of two independent parts. It is wise to make good use of your time.
- You should read the whole problem set before you start solving the problems.
- You can score a total of 100 points on this exam. The number of points stated on each part indicates the weight of that part.
- You can make your own clarifications if you find the examination text ambiguous or imprecise. Such clarifications must be written clearly in the delivered answer.
- Make short and clear explanations!

Good luck!

(Continued on page 2.)

Problem 1 Unisex Shower Room

Read the following description of a synchronisation problem:

> A group of men and women share a shower room. The men and women are exercising and want to take a shower after certain intervals of exercising and exercise after taking a shower. The synchronisation problem is, that there must never be a man and a woman in the shower room at the same time. The number of persons of the same gender in the shower room is, however, not limited.

Implement solutions to this problem as pointed out below. Use the special statement `shower` to indicate that a process is now taking a shower, and `exercise` when a process is not taking a shower. Consider the shower room to be the shared resource.

1a Implementation using semaphores (weight 10)

Implement a solution using *semaphores* for coordinating access to the shower room. Make sure, that the implementation is correct and that it does not contain deadlocks.

1b Correctness (weight 5)

Explain briefly, why your solution is safe, i.e., it is never the case that a man and a woman are in the shower room at the same time.

1c Deadlock freedom (weight 5)

Explain briefly, why your implementation is free of deadlocks.

1d Absence of starvation (weight 5)

Is your solution fair? If not, explain briefly how you can make it fair. Fairness means, that your solution can guarantee that any man/woman who wants to take a shower eventually will take a shower? You may assume that everyone that enters the shower will eventually exit the shower.

(Continued on page 3.)

1e Implementation using monitors (weight 10)

Now implement the shower room problem using a monitor for synchronising access to the shower room. The monitor should offer the following procedures

```
procedure enter(bool ismale){...}
procedure leave(bool ismale){...}
```

where the parameter `ismale` is true if the person who wants to enter is a man, and false if it is a woman.

Make sure, that the implementation is safe and that it does not contain deadlocks. The solution does not have to be fair. Which signalling discipline do the monitors in your solution use?

1f Specification (weight 5)

Consider the implementation from Problem 1e and define a suitable invariant that captures the safety property:

> At any given time it is never the case that a man and a woman is in the shower room at the same time.

1g Safety (weight 10)

Establish using program logic (Hoare logic) that your invariant from Problem 1f is maintained by the procedure "enter" in your implementation in Problem 1e.

1h Differences between semaphore and monitors (weight 5)

Explain briefly the differences between your solution using semaphores and the solution using monitors. Focus on the different methods for guaranteeing absence of deadlocks and on the different methods for ensuring fairness.

(Continued on page 4.)

Problem 2 Asynchronous communication

2a Shower by message passing (weight 10)

Make a solution to the shower problem above based on asynchronous message passing, using the language with **send** and **await** statements, and the non-deterministic choice statement `S1 [] S2`, which chooses either `S1` or `S2` for execution. The cycle of a person wishing to take a shower could be according to the following sketch:

```
send S:request(ismale); await S:enter; shower; send S:leave
```

where S refers to the shower agent (doorman) and *ismale* defines the gender (false for female, true for male). Your task is to program the shower agent S.

Note: If you need a data-structure for lists, say lists of agent references/names, you may assume that the expression $add(l, x)$ gives l with the element x appended (added at the end) and $remove(l, x)$ gives l with all occurrences of the element x removed, $first(l)$ gives the first element of l, and $rest(l)$ gives the rest of l without the first element. The empty list is represented by the constant *empty*.

2b Properties (weight 5)

Is your solution fair to both men and women? And if not, indicate briefly how to make it fair.

2c Alphabet (weight 5)

What is the alphabet of the histories of the shower controller S?

2d Specification of queues (weight 10)

Use the history of the shower controller to define the sequence of persons "inside" the shower and the sequence of persons "outside" the shower room (in the order they want to enter the shower room, respectively, entered it). A person can be considered inside the shower, when the controller has sent an enter message to the person but not yet received a leave message from that person. A person can be considered outside the shower, when the controller has received a request from the person but not yet sent the enter message to that person. More specifically, you should define two functions, $inside(ismale, h)$ and $outside(ismale, h)$, where *ismale* is the gender (true for male and false for female) and h is the local history of the shower controller S.

(Continued on page 5.)

2e Invariant (weight 5)

Formulate an invariant for the program above, ensuring that men and women are not showering at the same time. The invariant should be such that it can be verified for your implementation by itself, without additional conditions.

2f The shower agent in Creol (weight 5)

Program the shower agent *S* in the Creol language. The cycle of a person wishing to take a shower could be according to the following sketch:

```
await S.enter(ismale); shower; S!leave
```

where *S* refers to the shower object and *ismale* defines the gender (as above).

Hint: You may use lists with the above syntax. Note that the reserved word *caller* may be used inside a method to denote the caller object.

2g Robustness of the different solutions (weight 5)

The solutions will probably make assumptions on the way your synchronisation mechanism is used. Briefly discuss and compare how each solution will perform in the presence of misbehaving users, e.g., when one user does not make the expected call or action at the right time. Which of your solutions remain safe, i.e., avoid having a man and a woman in the shower at the same time? How do they react to abuse?