**UNIVERSITY OF HELSINKI**
**Faculty of Science**
**Dept. of Computer Science**

**Auvo Häkkinen**

**581332 Rinnakkaisohjelmistot**

**Separate Examination**
**20.01.2004**

Please write on each paper the date and the name of the course as well as your name and signature. Try to keep your answers short.

1 ONE-LANE BRIDGE (15p)

An island is connected to the mainland by a narrow bridge, where cars are allowed to drive only in one direction in a time. The car processes call procedure *enter_bridge(direction)* before using the bridge, and procedure *exit_bridge(direction)*, when thy leave the bridge. The parameter direction is the driving direction: “to the island” or “from the island”. The code for the car processes is

```
process car[1 to N] {
  ...
  enter_bridge(direction);
  drive on bridge;
  exit_bridge(direction);
}
```

Give the code for procedures *enter_bridge()* and *exit_bridge()*. The solution must be based on semaphores and must allow several cars on bridge driving to the same direction. The solution needs not to be fair, which means that the waiting times on the other end may be very long. When the waiting ends, the cars must be allowed to proceed in FCFS order.

2 ROLLER COASTER (15 p)

There are N customer processes and one car process. The passengers repeatedly wait to take rides in the car, which can hold C passengers ($C < N$). However, the car goes around the track only when it is full. When the car stops, all passengers have to leave the car. If they want to have another ride, they have to queue again.

Explain the conditions where synchronization and mutual exclusion is needed. Develop the code for the passenger and car processes. Use monitors. Explain how your solution handles the synchronization and mutual exclusion.

3 PARKING HALL (15 p)

There are several doors to a parking hall. Each door is controlled by a sensor(machine), that notifies when a car is coming in or going out. On each door, there is also a screen(machine) to show information to customers. The whole system is controlled by a central computer, which contains (among others)

- one process for each door, and

- one process to control all information screens.

The communication between the sensor and the door process is based on message-passing: the sensor gives a message if a car is coming in or going out. The door processes have a common counter indicating the number of free places in the hall. When the hall gets full, all screens start to show text "FULL". When there is room again, the screens show text "ROOM". The communication between the control process and the screens is also based on message passing: the control process sends the new text screens, when appropriate. The communication between the door processes and the controlling process is based on shared memory (efficiency!)

Give the essential parts of the code for door processes and the controlling process (i.e.: the variables needed and the communication code). You need not to explain how the sensor drivers and the screen drives would be implemented.

## 4 DEADLOCKS (15 p)

a) Consider a ticket reservation system, and give an example of each of the basic problems in the concurrent systems area (mutual exclusion, synchronization, deadlock, starving).

b) Which are the necessary conditions that must be present for a deadlock to be possible? How is each of these conditions fulfilled in the dining philosophers' problem? Would it be possible to prevent a deadlock in the dining philosophers' problem by changing the basic rules? How? Rethink each of the conditions.

c) Explain the basic ideas of the Banker's algorithm, and what kinds of data structures are needed to implement it. What happens to the requesting process, if the algorithm shows that the request can not be granted? What would happen, if the resource is granted, although according to the Banker's algorithm the request should be rejected?