THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2002

COMP2310
(Concurrent and Distributed Systems)

Writing Period: 3 hours duration

Study Period: 15 minutes duration

Permitted Materials: None

All your answers must be written in the boxes provided in this booklet. You will be provided with scrap paper for working, but only those answers written in this booklet will be marked. Do not remove this booklet from the examination room. There is additional space at the end of the booklet in case the boxes provided are insufficient. Label any answers you write at the end of the booklet with the number of the question they refer to.

Greater marks will be awarded for answers that are simple, short and concrete than for answers of a sketchy and rambling nature. Marks will be lost for giving information that is irrelevant to a question.

Name (family name first):

Student Number:

Official use only:

Q1 (25)	Q2 (25)	Q3 (25)	Q4 (25)	Q5 (25)	Q6 (25)	Total (150)

QUESTION 1 [25 marks]

(a) 'An operating system is an example of an inherently concurrent system.' Make an argument to support this assertion.

QUESTION 1(a) **[4 marks]**

(b) Define the term *MIMD* and give an example of a *MIMD* computer.

QUESTION 1(b) **[3 marks]**

(c) Define the terms *process* and *thread*, being careful to distinguish them.

QUESTION 1(c) **[4 marks]**

(d) Describe what happens when the `exec()` system call is invoked.

QUESTION 1(d) **[4 marks]**

(e) Define the *mutual exclusion* problem in detail, including a list of the properties a solution to the problem must have.

QUESTION 1(e) **[6 marks]**

(f) If a module is a *monitor*, what property does it have? Explain in detail.

QUESTION 1(f) **[4 marks]**

QUESTION 2 [25 marks]

(a) (i) One of the standard capabilities of a Unix shell is to interpret command lines of the form:

$$c_1 \mid c_2 \mid \ldots \mid c_n$$

where each c_i is a command consisting of a program name followed by the arguments for the program. Give a useful example of such a pipeline, made up of three or four commands, and explain what it does. Use only standard Unix programs, such as `grep`.

QUESTION 2(a)(i) **[3 marks]**

(ii) Explain how the shell program causes such pipelines to be formed. In doing so, show how the library calls `fork()`, `exec()`, and `wait()` are used. Also explain (again, with reference to the necessary library calls) how the shell directs the flow of data from each stage of the pipeline to the next.

QUESTION 2(a)(ii) **[12 marks]**

QUESTION 2(a)(ii) continued

(b) Implement a monitor to manage a bounded buffer. Allow for multiple producers (which call monitor procedure `Insert`) and multiple consumers (which call monitor procedure `Remove`). In your implementation, include the statements required to insert and remove the data items from the buffer. (Do *not* implement the behaviour required in the assignment, where *all* consumers were required to see each item that had been inserted into the buffer.) You may use either *Signal and Continue* or *Signal and Urgent Wait*; specify which one you are using.

QUESTION 2(b) **[10 marks]**

QUESTION 3 [25 marks]

(a) Explain the behaviour of the operations `wait` and `signal` that belong to the semaphore abstract data type.

QUESTION 3(a) **[5 marks]**

(b) The Fetch-and-Add instruction is defined (using a C-like syntax) as follows:

```
int FA(int var,incr) {
  int tmp = var; var = var + incr; return tmp;
}
```

The value of `incr` may be negative. Of course, the body is executed atomically. Use the `FA` instruction to implement semaphores. You will need to use busy-waiting in your implementation of `wait`.

QUESTION 3(b) **[5 marks]**

(c) Semaphores and locks can each be used to achieve mutual exclusion. Contrast the two, in terms of CPU utilization, when used for this purpose.

QUESTION 3(c) **[5 marks]**

(d) In the context of distributed systems, define *scalability* and *transparency*.

QUESTION 3(d) **[2 marks]**

scalability:

transparency:

(e) Threads can either be implemented at the user level or at the kernel level. Explain what difference this makes, especially with regard to how system calls behave.

QUESTION 3(e) **[3 marks]**

(f) Suppose we have a process creation primitive `xfork()`. It is similar to the UNIX process creation primitive `fork()`: it returns some kind of process identification for the child process to the parent, and it returns 0 to the child. However, we do not know whether it creates a heavyweight process (like `fork` does) or a thread. Devise a test (write a C program) to determine which is the case, and explain how it works and how its output should be interpreted. You may also want to make use of the primitives `xwait(pid)`, which waits for the process/thread with identification `pid` to terminate, and `xexit()`, which terminates a process/thread.

QUESTION 3(f) **[5 marks]**

QUESTION 4 [25 marks]

(a) Define each of the following terms related to message passing: unicast, multicast, broadcast, message timeout, message expiration.

QUESTION 4(a) **[5 marks]**

unicast:

multicast:

broadcast:

message timeout:

message expiration:

(b) Explain in your own words what a *future* is. With the aid of an example, show how a future can be implemented in Linda.

QUESTION 4(b) **[3 marks]**

(c) Define the term ‘connection-oriented protocol’. Give an example of a standard network protocol which is connection-oriented.

QUESTION 4(c) **[3 marks]**

(d) What is Lamport’s Algorithm? Describe (a) what it is for, and (b) how it works.

QUESTION 4(d) **[5 marks]**

(e) What is the motivation for providing an RPC facility? What are its advantages over message passing?

QUESTION 4(e) **[3 marks]**

(f) Define the term ‘election algorithm’.

QUESTION 4(f) **[1 mark]**

(g) Describe the Bully Algorithm with the aid of a small example.

QUESTION 4(g) **[5 marks]**

QUESTION 5 [25 marks]

(a) For Ricart and Agrawala's distributed algorithm for distributed mutual exclusion, identify (1) the facilities that must be provided for it to work; (2) the number of messages that must be sent for each critical region entry (as a function of n, the total number of processes); (3) how to deal with lost messages. Using an example, show how the algorithm works.

QUESTION 5(a) **[10 marks]**

facilities required:

number of messages:

lost messages:

example:

(b) Define the terms 'safety property' and 'liveness property', and give an example of each.

QUESTION 5(b) **[4 marks]**

safety:

liveness:

(c) What is distributed deadlock detection? (Distinguish it from deadlock prevention, etc.)

QUESTION 5(c) **[3 marks]**

(d) Give a definition of the terms flat transaction, nested transaction, and distributed transaction.

QUESTION 5(d) **[3 marks]**

flat:

nested:

distributed:

(e) What is a serialization graph?

QUESTION 5(e) **[2 marks]**

(f) Under what condition is a transaction history serializable?

QUESTION 5(f) **[1 mark]**

(g) Describe what happens when a subtransaction of a nested transaction aborts.

QUESTION 5(g) **[2 marks]**

QUESTION 6 [25 marks]

(a) Describe, with the aid of an appropriate diagram, how Java RMI works. Your description should include (at least) references to the stub, skeleton, and registry.

QUESTION 6(a) **[10 marks]**

(b) Describe, with the aid of an appropriate diagram, how MPI can be used to implement the centralized algorithm for distributed mutual exclusion. Your description should include the various MPI calls and the tags you use, but don't write any code. Show how to use your solution to solve the Dining Philosopher's problem.

QUESTION 6(b) **[10 marks]**

(c) Both Java and .NET support *object serialization*. Explain what this is, and what role it plays in building distributed systems.

QUESTION 6(c) **[5 marks]**

Additional answers. Clearly indicate the corresponding question and part.

Additional answers. Clearly indicate the corresponding question and part.