The Australian National University
Final Examination – November 2010

COMP 2310 / COMP 6310 Concurrent and Distributed Systems

Study period:	15 minutes
Time allowed:	3 hours
Total marks:	100
Permitted materials:	None

Questions are not equally weighted – sizes of answer boxes do not necessarily relate to the number of marks given for this question.

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 answer you write at the end of the booklet with the number of the question it refers to.

Give precise, technical answers, and read the questions carefully. Guessing is discouraged. If you do not know the answer to a question, try to derive at least a partial answer based on your knowledge, while providing intermediate steps to illustrate your thought process.

Student number:

The following are for use by the examiners

Q1 mark	*Q2 mark*	*Q3 mark*	*Q4 mark*	*Q5 mark*	*Q6 mark*	*Total mark*

1. [10 marks] Concurrency

(a) [3 marks] Provide examples for events that can cause a process state to change

- from ready to running.
- from running to ready.
- from running to blocked.
- from blocked to ready.

(b) [7 marks] You are asked to design a new programming language in which the compiler can automatically create concurrent code that can fully utilise multi-core CPUs, without the need for the programmer to explicitly define concurrency in the code. Which concepts / primitives / paradigms would you suggest to include in your new language? Name at least two, and give details why, how and under what circumstances they allow the compiler to create concurrent code.

2. [14 marks] Synchronisation

(a) [6 marks] In Ada protected objects, what is the difference between the requeue statement and a procedure call? Explain why the requeue statement is necessary, and provide a simple example.

(b) Assume three tasks T1, T2 and T3 consisting of three distinct, sequential blocks of operations $A_i \rightarrow S_i \rightarrow B_i$ (each task first executes its A_i, then S_i, then B_i). Provide synchronisation code S_i for each task so that no B_i starts execution before all A_i are complete. You may use pseudo-code, but be precise. Use the following primitives to implement synchronisation:

(i) [4 marks] Asynchronous message passing: implement S_1, S_2 and S_3 using

```
procedure send(to_task, message) -- sends an asynchronous message to the specified task

function receive() return message -- returns a received message, blocks if there is no message
```

(ii) [4 marks] Synchronous message passing: implement S_1, S_2 and S_3 using task entries as provided by Ada.

3. Scheduling [8 marks]

```
Task T1;
Task body T1 is
begin
   loop
       compute_for_2_time_units;
       T2.signal;
   end loop;
end;
```

```
Task T2 is
    entry Signal;
end;
Task body T2 is
begin
   loop
      compute_for_3_time_units;
      accept Signal;
   end loop;
end;
```

(a) [4 marks] Schedule T1 and T2 using Round Robin. (Each time slice is one time unit long.) Task T1 starts 3 time units after Task T2. Show the first 10 time units.

[4 marks] Schedule T1 and T2 using First-Come-First-Serve. Task T1 starts 3 time units after Task T2. Show the first 10 time units.

4. [12 marks] Nondeterminism

Consider the following three versions of a task:

```
task selector is  -- (identical)
    entry E1;
    entry E2;
end;
```

```
-- (I) --
   task body Selector is
   begin
      for I in 1..2 loop
         select
            accept E1 do
               delay 1.0;
               Put (“1”);
            end;
         or
            accept E2;
            delay 1.0;
            Put (“2”);
         end select;
         Put(“3”);
      end loop;
   end;
```

```
-- (II) --
   task body Selector is
   begin
      for I in 1..2 loop
         select
            accept E1 do
               delay 1.0;
               Put (“1”);
            end;
         or
            delay 1.0;
            accept E2;
            Put (“2”);
         end select;
         Put(“3”);
      end loop;
   end;
```

```
-- (III) --
   task body Selector is
   begin
      for I in 1..2 loop
         select
            accept E1 do
               delay 1.0;
               Put (“1”);
            end;
         else
            accept E2;
            delay 1.0;
            Put (“2”);
         end select;
         Put(“3”);
      end loop;
   end;
```

```
-- main program
begin
   delay 2.0;
   select Selector.E1;
      Put (“A”);
   else
      Put (“X”);
   end select;
   delay 1.0;
   Selector.E2;
   delay 0.5;
   Put(“B”);
end;
```

(a) [8 marks] Provide a time line of the program's output for version (I), (II) and (III) of task body Selector. Indicate (in seconds) when certain outputs occur.

(b) [4 marks] Which version(s) of the program will not terminate, and why not?

5. Safety and Liveness [34 marks]

(a) [4 marks] What are the necessary conditions for a deadlock?

(b) [6 marks] What is the difference between Deadlock Avoidance and Deadlock Prevention? Describe briefly how to prevent deadlocks.

(c) The Banker's algorithm can be used to detect deadlocks that are caused by multiple transactions holding locks on shared objects.

(i) [4 marks] What information about current transactions needs to be available for the Banker's algorithm to work?

(ii) [10 marks] If the Banker's algorithm declares the system as "safe" given the current state, and assuming that there will not be any new transactions, does this mean that there cannot be any deadlocks involving the currently active transactions? If the Banker's algorithm declares that the system is not safe, does this mean that there will definitely be a deadlock? Justify your answers.

(iii) [4 marks] Describe how the Banker's algorithm can be used to avoid deadlocks.

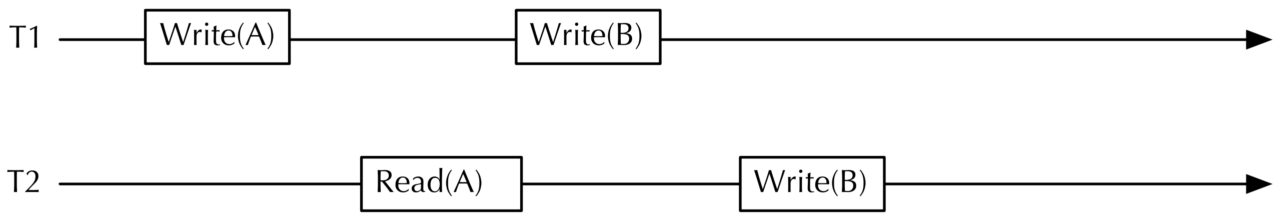
(d) [6 marks] A customer wants you to design a "fail safe" control system for a pilotless robotic aircraft. What is a possible system behaviour that would need to be implemented to fulfil the customer's request? What would you suggest if the customer asked you for an alternative approach?

6. Distributed Systems [22 marks]

(a) [2 marks] What are the necessary and sufficient conditions for two transactions to be serializable?

(b) [2 marks] What are the four "ACID" properties for transactions?

(c) [6 marks] Two transactions T1 and T2 have conflicting pairs of operations on shared data objects A and B. Both transactions are executed concurrently on the shared objects, while making sure that serializability is maintained at all times. Transaction T1 encounters an error and aborts. How does this potentially affect T2? Give details on possible scenarios.



[12 marks] The snapshot algorithm was introduced as a way to obtain a consistent snapshot of the system state in a distributed system. One problem for the process collecting the snapshot is how to decide when the snapshot is complete. Explain why this is a problem, and describe in detail how to decide when the snapshot is complete without making assumptions about message delays.

continuation of answer to question ☐ *part* ☐

continuation of answer to question ☐ *part* ☐

continuation of answer to question | | *part* | |

continuation of answer to question | | *part* | |

continuation of answer to question | | *part* | |

continuation of answer to question | | *part* | |

continuation of answer to question | | *part* | |

continuation of answer to question | | *part* | |