THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester Examination, November 2007

COMP2310
(Concurrent and Distributed Systems)

Writing Period: 3 hours duration

Study Period: 15 minutes duration
Permitted Materials: Non-Programmable Calculator

Answer all questions

The questions are followed by labelled, framed blank panels into which your answers are to be written. Additional answer panels are provided (at the end of the paper) should you wish to use more space for an answer than is provided in the associated labelled panels. If you use an additional panel, be sure to indicate clearly the question and part to which it refers to.

More marks are likely be awarded for answers that are short and concrete than for answers of a sketchy or rambling nature. Answers which are not sufficiently legible may not be marked.

Depending on your relative performance in the mid-semester exam, and in accordance with the assessment scheme given on the course web page, this exam will represent either 50% or 60% of your total course mark.

Name (family name first):

Student Number:

Official use only:

Q1 (8)	Q2 (18)	Q3 (10)	Q4 (10)	Q5 (6)	Q6 (14)	Q7 (13)	Q8 (21)	Total (100)

QUESTION 1 [8 marks] General Concurrency and its Support

(a) Detail what events might cause the following process transitions:

i. from running state to blocked state

ii. from running state to ready state

iii. from blocked state to ready state

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

(b) A concurrent language must provide some concept of a task/thread or other concurrent entity. List TWO other elements that relate to concurrency and might be provided by a concurrent language.

QUESTION 1(b)	**[2 marks]**

(c) Give a precise definition of an atomic operation **AND** identify **two** different atomic operations that might be supported in hardware on a modern computer system.

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

QUESTION 2 [18 marks] Mutual Exclusion and Synchronization

(a) In the context of concurrent programming explain what is meant by a race condition? Include in your answer 20 lines or less of pseudo code that shows a race condition.

QUESTION 2(a) **[4 marks]**

(b) Both conditional critical regions and monitors enforce mutual exclusion. Why are monitors a more efficient way to implement mutual exclusion in a large application code?

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

Question 2 (continued)

(c) The following Ada code creates three tasks that each increment a local counter and prints out its value in an infinite loop. The code compiles and runs correctly (until the value of the counters exceed the maximum value for the integer data type). Outline how you would modify this code such that at any given time:

- P_Count > Q_Count
- P_Count + Q_Count > R_Count
- All counters can reasonably expect to increment over time at a non-zero rate

You are free to use any Ada2005 functionality. You are not required to give the exact Ada2005 syntax, providing that you make your intentions clear. You will be marked both on the correctness of your approach and its likely runtime efficiency.

```
with Ada.Text_IO; use Ada.Text_IO;
procedure PQR is
   task P;
   task Q;
   task R;

   task body P is
      P_Count : Integer := 0;
   begin
      loop
         P_Count := P_Count + 1;
         Put_Line("Value of P " & P_Count'img);
         delay 0.0;
      end loop;
   end P;

   task body Q is
      Q_Count : Integer := 0;
   begin
      loop
         Q_Count := Q_Count + 1;
         Put_Line("Value of Q " & Q_Count'img);
         delay 0.0;
      end loop;
   end Q;

   task body R is
      R_Count : Integer := 0;
   begin
      loop
         R_Count := R_Count + 1;
         Put_Line("Value of R " & R_Count'img);
         delay 0.0;
      end loop;
   end R;
begin
   null;
end PQR;
```

QUESTION 2(c) **[8 marks]**

(d) The following class attempts to implement a general semaphore in Java. It contains **two** different logical errors. Identify the **two** errors and correct them. For each logical error state what the effect would be if the code contained only that error. (Note: the fact that `value` can become negative should not be considered as one of the two errors.)

```
class semaphore {

  protected int value = 0 ;

  protected semaphore() { value = 0 ; }

  protected semaphore(int initial) { value = initial ; }

  public synchronized void P() {
    value = value + 1;
    if (value < 0)
      try { wait() ; } catch( InterruptedException e ) { }
  }

  public synchronized void V() {
    value = value - 1; if (value <= 0) notifyall() ;
  }
}
```

QUESTION 2(d)	**[4 marks]**

Student Number: ....

QUESTION 3 [10 marks] Message Passing

(a) Consider the Ada code outlined below:

```
task T1;

task body T1 is
begin
   ...
   T2.synch;
   ...
end T1;
```

```
task T2;
  entry synch;
end T2;

task body T2 is
begin
   ...
  accept synch do;
    T3.synch;
  end synch;
   ...
end T2;
```

```
task T3
  entry synch;
end T3;

task body T3 is
begin
   ...
  accept synch;
   ...
end T3;
```

(i) Explain how the Ada rendezvous has been used to achieve synchronizations between all three tasks.

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

(ii) Describe what the consequences would be if the accept statement of T2 was changed to read as follows:

```
 ...
 accept synch;
T3.synch;
 ...
```

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

Question 3 (continued)

(b) Detail two aspects of a remote procedure call (or remote method invocation) that are significantly harder for the underlying environment to deal with compared to a local procedure call.

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

(c) In what way is the Ada rendezvous construct asymmetrical?

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

QUESTION 4 [10 marks] Non-Determinism

(a) Describe two scenarios where it would be beneficial to employ an Ada `requeue` statement.

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

Question 4 (continued)

(b) The following shows a series of interactions between 4 Ada tasks. Within each task important lines have been numbered. Your objective is to show whether the interactions between these 4 tasks proceed in a deterministic or non-deterministic fashion and whether they do/could give rise to deadlock. You should analyse the code by using a tuple of the form (ABCD) to indicate which line each task is currently positioned at and then gives arrows to indicate possible transitions from that state to another. You should assume that the tasks begin in state (1111). You are free to augment this representation as you see fit, providing that you make clear what you have done. The first transition is from (1111)→(2112).

```
Line  Task A

      loop
 1     accept A1;
 2     accept A2;
      end loop;
      end A;
```

```
Line  Task B

      loop
 1     select
 2      accept B1 do
 3       A.A1;
        end B1;
       or
 4      accept B2 do
 5       A.A2;
        end B2;
       end select;
      end loop;
      end B;
```

```
Line  Task C

      loop
 1     A.A2;
 2     accept C1 do
 3      A.A1;
 4      B.B2;
       end C1;
 5     D.D1;
      end loop;
      end C;
```

```
Line  Task D

      loop
 1     A.A1;
 2     C.C1;
 3     accept D1 do
 4      B.B1;
 5      A.A2;
       end D1;
      end loop;
      end D;
```

QUESTION 4(b) **[6 marks]**

QUESTION 5 [6 marks] Scheduling

(a) Suppose processes P_1, P_2 and P_3 have computation times of 6, 2 and 1 units and become ready to run at times 0, 1 and 2, respectively. With the aid of a diagram, illustrate how these will be scheduled under the First-Come-First-Serve (FCFS) and Round-Robin (RR) scheduling policies. What is (are) the principal advantage(s) of RR in this situation?

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

(b) In a more general context, what is the main disadvantage of Round-Robin scheduling? If the process computation times are known in advance, briefly describe a policy which suffers less from this problem, but is still free of starvation.

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

Student Number: ....

QUESTION 6 [14 marks] Safety and Liveness

(a) List two important safety properties in concurrent systems.

QUESTION 6(a) **[2 marks]**

(b) Fairness as a means to avoid starvation is a classical liveness property. In this context, explain the difference between the *linear waiting* (LW) and *first-in, first-out* (FIFO) schemes. Which of these two schemes would you be more likely to see in distributed systems? Briefly explain.

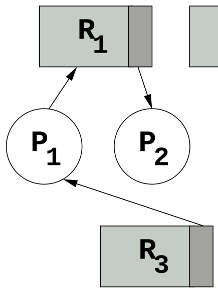
QUESTION 6(b) **[4 marks]**

Question 6 (continued)

(c) Briefly describe the four necessary conditions for deadlock in a concurrent system.

QUESTION 6(c)	**[4 marks]**

(d) Consider the following resource allocation graph involving processes P_1, P_2 and P_3 and resources R_1, R_2 and R_3.



(i) Explain in words the above diagram.

QUESTION 6(d)[i] **[2 marks]**

(ii) Assuming all currently held resources are maintained, give two resource requests that would result in an immediate deadlock.

QUESTION 6(d)[ii] **[1 mark]**

(iii) Assuming all currently held resources are maintained, give two resource requests that would not result in an immediate deadlock.

QUESTION 6(d)[iii] **[1 mark]**

QUESTION 7 [13 marks] Architectures

(a) List a 'minimal' set of (at least 4) features / services that any operating system would be expected to support.

QUESTION 7(a) **[2 marks]**

(b) Briefly describe the monolithic and micro-kernel approaches to operating systems design. Give one advantage and one disadvantage of the former over the latter.

QUESTION 7(b) **[3 marks]**

(c) Both pipes and sockets provide a byte-stream abstraction for inter-process communication. Aside from possible differences in performance, briefly describe three major advantages (or disadvantages) of one over the other.

QUESTION 7(c) **[3 marks]**

(d) Describe how you would write a (C) program which implements the Unix compound command `ls -l | wc`. You may do this by writing pseudo-code. *Hint:* use the system calls pipe(), fork(), execlp(**char** *file, **char** *arg, ...) and dup2(**int** oldfd, **int** newfd).

QUESTION 7(d) **[5 marks]**

QUESTION 8 [21 marks] Distributed Systems

(a) Define the term *idempotent operation*. Give an example of where the idempotency of an operation might be desirable in the presence of faults (e.g. communication failures) in a distributed system. Briefly explain why.

QUESTION 8(a) **[3 marks]**

(b) This question addresses issues with virtual (logical) times in distributed systems.

(i) If you find two logical times $C(a)$ and $C(b)$ associated to events a and b in different processes, what can you conclude if (1) $C(a) = C(b)$, and (2) $C(a) \leq C(b)$.

QUESTION 8(b)[i] **[2 marks]**

(ii) Alternatively, if you know something about the the relationship between events a and b in a distributed system, what can you conclude about their virtual times if (1) a happened concurrently with b, and (2) a is the send event and b is the receive event for the same message.

QUESTION 8(b)[ii] **[2 marks]**

(iii) In any implementation of virtual times, the sequence of timestamps seen by any process should never decrease. Briefly explain what problem may arise if a later timestamp has a lower value than an earlier one.

QUESTION 8(b)[iii] **[1 mark]**

(iv) The three general strategies for dealing with deadlock are detection / recovery, avoidance and prevention. Consider these stategies in the context of transactions.

(i) What property of transactions makes detection / recovery an acceptable strategy?

QUESTION 8(b)[i] **[1 mark]**

(ii) Which strategies are applicable to the two-phase locking, time-stamp ordering (TSO) and optimistic transaction schemes?

QUESTION 8(b)[ii] **[3 marks]**

Question 8 (continued)

(iii) Compare the strengths and weaknesses of the two-phase locking, time-stamp ordering (TSO) and optimistic schemes.

QUESTION 8(b)[iii]	**[3 marks]**

(iv) Describe how virtual (logical) clocks may be used to implement *either* distributed critical regions *or* the consistent collection of distributed states (by an observer process).

QUESTION 8(b)[iv]	**[6 marks]**

Additional answers to QUESTION ___(___)[___]

Additional answers to QUESTION ___(___)[___]

Additional answers to QUESTION ___(___)[___]

Student Number: ....

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]