THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester Examination – November 2004

COMP2310

Concurrent and Distributed Systems

Study Period: 15 minutes

Time Allowed: 3 hours

Permitted Materials: None

Questions are NOT equally weighted.

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:

The following are for use by the examiners.

Q1 Mark	Q2 Mark	Q3 Mark	Q4 Mark	Q5 Mark	Q6 Mark	Total Mark

Student Number: ....

Question 1 [10 marks] General Concurrency

(a) In a technical system under what conditions are two events considered concurrent?

[2 marks]

(b) What is the general definition of a process? In the context of an operating system what attributes are usually attached to a process?

[2 marks]

(c) Processes in Unix are created using "`fork`". Why is this technique problematic in terms of efficiency, and how is this issue resolved in current Unix implementations?

[2 marks]

(d) In the context of an operating system sketch all process states and their possible transitions (incl. states referring to secondary memory).

[4 marks]

Student Number: ....

Question 2 [25 marks] Synchronization

(a) You are responsible for a software development team that is writing a large application to be run in a concurrent and potentially distributed environment. The application requires the creation of a shared counter (`N`) and several child tasks (`Plus_One`). After creation each child task is required to increment the shared counter, reporting its value immediately before and after being incremented. The main thread of execution is required to report the value of the shared counter before it has been modified by *any* child task and after it has been incremented by *all* child tasks. Your design team implement this aspect of the application within a single package (`Sightings`). All parts of this package that relate to task creation and manipulation of the shared counter are given below. (The package, as given below, is syntactically correct and will pass the Ada95 compiler without any warnings. The call "`Image (Current_Task)`" delivers a string containing the current Task_Id, which is then used in the output message to identify the task).

```
--- Specification part of Sightings ---
package Sightings is

  No_Of_Tasks : constant Positive := 5;
  N           :          Natural  := 0;

  procedure Observe;

end Sightings;

--- Implementation part of Sightings ---
with Ada.Text_IO;             use Ada.Text_IO;
with Ada.Task_Identification; use Ada.Task_Identification;

package body Sightings is

  procedure Observe is

     task type Plus_One;

     task body Plus_One is

        begin
           Put_Line ("N   in task " & Image (Current_Task) & " is:" & N'img);
           delay 0.0;
           N := N + 1;
           Put_Line ("N+1 in task " & Image (Current_Task) & " is:" & N'img);
        end Plus_One;

     Plus_One_Tasks : array (1..No_Of_Tasks) of Plus_One;

  begin
     Put_Line ("N in procedure Observe (beginning) :" & N'img);
     delay 0.0;
     Put_Line ("N in procedure Observe (end)       :" & N'img);
  end Observe;

end Sightings;
```

(i) Would you consider this package and its use of the shared counter `N` to be a good and correct solution to the program requirements as detailed above? Explain your answer technically and precisely. If you are not satisfied with this solution, suggest a better alternative. (You are not required to write a new Ada package - just to describe what you would like to change/introduce/delete).

(i)

[4 marks]

(ii) Regardless of your recommendation given in part (i), this code is already deployed in an actual installation. Thus you turn your attention to considering all possible output scenarios. Detail under what circumstances what output will be obtained. If in order to do this you require further information, detail exactly what information you require and why.

(ii)

[8 marks]

(b) Construct a mutual exclusion system based on atomic test-and-set operations for n processes. Choose any pseudo code form you like and discuss the features of your solution.

[4 marks]

(c) What are the differences between a binary semaphore and a test-and-set operation? Specify both constructs precisely before you discuss their differences.

[2 marks]

(d) What is a side-effect free operation and why and how can you make use of such operations in a concurrent system? Be as precise as possible. Which languages are intrinsically (mostly) side-effect free?

[3 marks]

(e) Compared to using semaphores as a synchronization primitive, which programming errors are prevented by construction when using protected objects (as supplied in Ada95)? Which programming errors are still possible?

[4 marks]

Student Number: ....

Question 3 [20 marks] Message Passing

(a) While setting up a message passing system between two networked computers, you observe that the message received is different from the message sent - even though you used the same programming language on both sides.

(i) What possible causes are there for this difference? Distinguish between reproducible and stochastic differences, complete (same length) and an incomplete (different length) message receipt, and message differences that occur immediately compared to those that appear only after some time.

(i)

[5 marks]

(ii) For each of the potential causes that you outlined in part (i) indicate how you would modify your program to test for this problem, and how you would try to rectify it.

(ii)

[8 marks]

(b) Support for interprocess communication (IPC) using `pipes` is provided by all flavors of Unix.

(i) To which general communication mechanism do Unix `pipes` belong?

(i)

[1 mark]

(ii) Can Unix `pipes` be applied to all possible communication scenarios in concurrent and distributed systems? If not, then what are their key limitations and what alternative method of Unix IPC would you use if these limitations were a problem?

(ii)

[3 marks]

(c) How would you emulate asynchronous message passing if your underlying communication system only supported synchronous message passing?

[3 marks]

Student Number: ....

Question 4 [6 marks] Scheduling

(a) In scheduling schemes what is measured by utilization tests and by response time analysis?

[2 marks]

(b) Could a feedback scheduling scheme with exponentially growing pre-emption intervals lead to starvation? Explain your answer exactly. Does it make a difference whether the pre-emption intervals grow exponentially or linearly?

[4 marks]

Question 5 [14 marks] Safety and Liveness

(a) What is deadlock prevention? Name possible strategies for implementing this.

[2 marks]

(b) What is deadlock avoidance? Name possible strategies for implementing this.

[2 marks]

(c) The following Ada program is syntactically correct and will compile without errors or warnings:

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Synchronize_It is

  task Stack_1 is
     entry Push;
     entry Pop;
  end Stack_1;

  task Stack_2 is
     entry Push;
     entry Pop;
  end Stack_2;

  protected Stack_3 is
     entry Push;
     entry Pop;
  private
     Filled : Boolean := False;
  end Stack_3;

  task Pop_Push;
  task Push_Pop;
```

```
task body Stack_1 is
begin
   loop
      select
         accept Push;
      or
         accept Pop;
      end select;
   end loop;
end Stack_1;

task body Stack_2 is
begin
   loop
      accept Push;
      accept Pop;
   end loop;
end Stack_2;

protected body Stack_3 is

   entry Push when not Filled is begin
      Filled := True;
   end Push;

   entry Pop when Filled is begin
      Filled := False;
   end Pop;

end Stack_3;
```

```
   task body Pop_Push is
   begin
      Stack_1.Pop; Stack_1.Push; Put_Line ("Pop_Push done on Stack_1");
      Stack_2.Pop; Stack_2.Push; Put_Line ("Pop_Push done on Stack_2");
      Stack_3.Pop; Stack_3.Push; Put_Line ("Pop_Push done on Stack_3");
   end Pop_Push;

   task body Push_Pop is
   begin
      Stack_1.Push; Stack_1.Pop; Put_Line ("Push_Pop done on Stack_1");
      Stack_2.Push; Stack_2.Pop; Put_Line ("Push_Pop done on Stack_2");
      Stack_3.Push; Stack_3.Pop; Put_Line ("Push_Pop done on Stack_3");
   end Push_Pop;

 begin
   null;
 end Synchronize_It;
```

Student Number: ....

Consider the tasks `Pop_Push` and `Push_Pop`: which of them will terminate? If one or both of them will not terminate, how far do you expect them to get (i.e. what output do you expect to appear)? If you think you need to distinguish multiple cases then describe each case precisely. Is your answer dependent on the underlying machine architecture (single-processor, multi-processor) or operating system?

[10 marks]

Question 6 [25 marks] Distributed systems

(a) Which kinds of nodes can be found in a network? Describe the different kinds in terms of the OSI network layer protocol.

[3 marks]

(b) Precisely what distinguishes a deterministic from a non-deterministic network architecture? Give examples of both.

[3 marks]

(c) Which of the ACID properties restricts the effectiveness of concurrent and distributed systems? In what sense can this property be "bent" or "violated" in order to achieve higher performance?

[2 marks]

(d) Transaction schedulers based on (strict) time-stamp ordering are said to be a good choice for distributed systems. Give reasons which support this statement or which could support its negation. Is (strict) time-stamp ordering based scheduling deterministic? Is this good or bad? Explain.

[6 marks]

(e) Compare implementations of distributed critical regions based on synchronized clocks with those based on logical clocks. Explain the advantages and drawbacks of both.

[5 marks]

(f) Taking a global snapshot of a distributed system requires special care to ensure that it does not result in an inconsistent snapshot. Assuming that you have no control over the snapshot algorithm itself, what would you request to be included in the snapshot state (by each involved process) so that you can decide afterwards whether the snapshot is consistent or not?

[6 marks]

Student Number: ....

Continuation of answer to Question ☐ Part ☐

Continuation of answer to Question ☐ Part ☐

Student Number: ....

Continuation of answer to Question ☐ Part ☐

Continuation of answer to Question ☐ Part ☐