

The Australian National University
Second Semester Examination – November 2008

COMP 2310

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.

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

<i>Q1 mark</i>	<i>Q2 mark</i>	<i>Q3 mark</i>	<i>Q4 mark</i>	<i>Q5 mark</i>	<i>Q6 mark</i>	<i>Total mark</i>

1. [7 marks] General Concurrency

- (a) [4 marks] Name the two basic forms of information exchange between processes in concurrent systems. Under which circumstances would you prefer one of these forms over the other (and visa versa)? (give examples)

- (b) [3 marks] Which events can cause processes to be transferred between the following scheduling states:

- (i) from 'created' to 'ready'
- (ii) from 'ready' to 'running'
- (iii) from 'running' to 'terminated'

2. [10 marks] Synchronization

(a) [2 marks] What is a semaphore? Give a precise definition.

(b) [4 marks] Semaphores are considered inadequate for large scale concurrent systems. Give at least two technically precise reasons for this statement.

- (c) [4 marks] Compare monitors and conditional critical regions as means to provide efficient and non-error prone synchronization between processes. Explain precisely what the differences are.

3. [6 marks] Message Passing

- (a) [3 marks] Can synchronous and asynchronous message passing systems emulate each other? For both cases: provide a sketch of an implementation and its limitations, or a reason why it cannot emulate the other message passing system.

- (b) [3 marks] Depending on the individual machine architectures on the sending and receiving computers, as well as the architecture of the communication system, conversion of the message contents will become necessary. Where and how is this implemented or provided?

4. [6 marks] Scheduling

- (a) [3 marks] You are requested to select or implement a scheduler for a practical system. Which questions would you ask before starting your design or selection? Provide at least five questions.

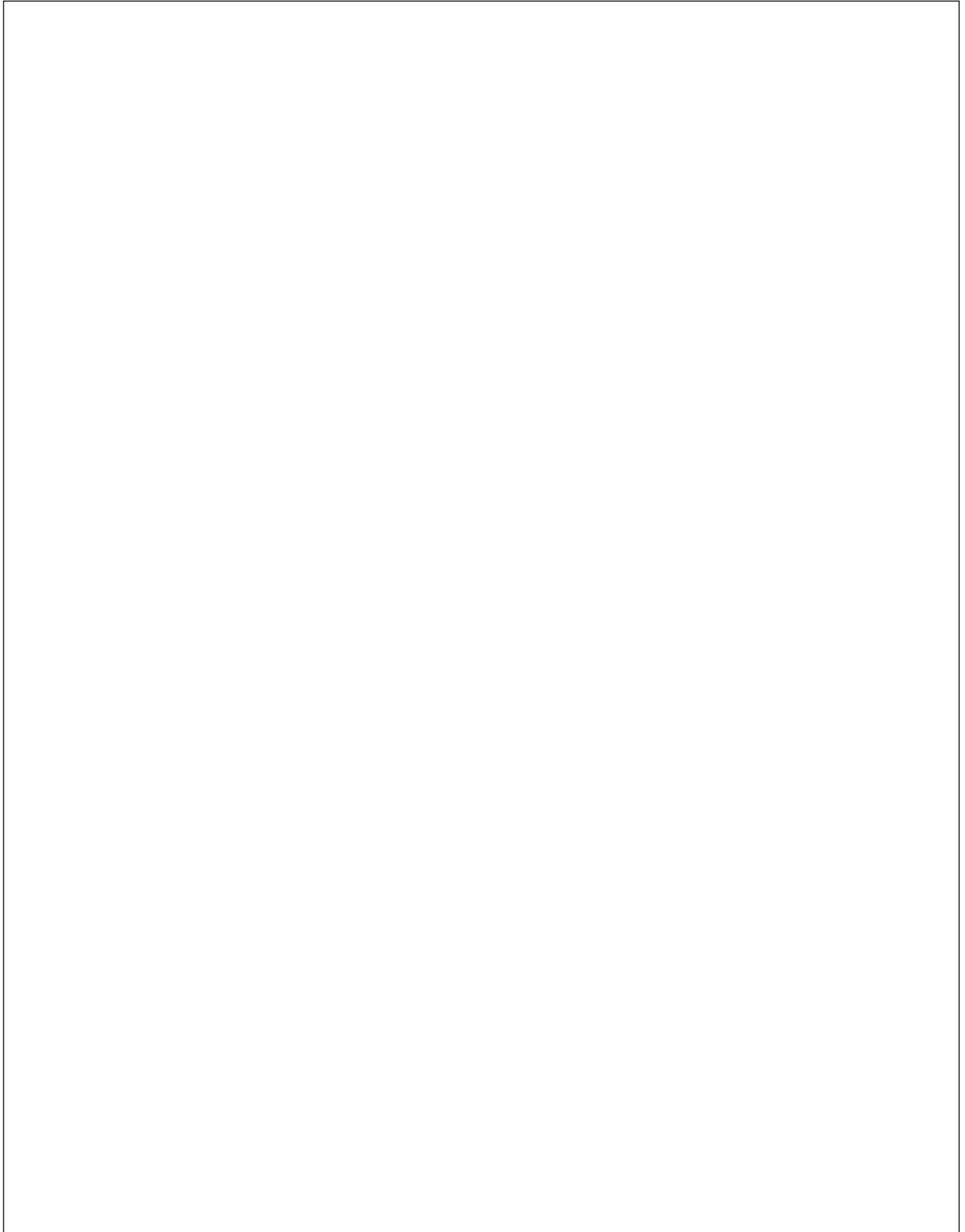
- (b) [3 marks] What specifies an optimal fixed priority scheduling scheme?

5. [33 marks] Safety and Liveness

- (a) [3 marks] Discuss the differences between a 'fault tolerant' and a 'fail safe' system. Provide a real-world system example which recommends itself for 'fault tolerant' and one example which recommends itself for a 'fail safe' design. Give reasons.

(b) [8 marks] Deadlock strategies

(i) [4 marks] What is required to implement a 'Deadlock prevention' strategy? Discuss multiple possibilities.



(ii) [4 marks] Under which practical circumstances would a 'Deadlock avoidance' strategy be preferable?

- (c) [22 marks] Read the following first part of an Ada program carefully. The whole program (the second part will be presented later in this question) is syntactically correct and will compile without warnings.

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Synced_Processes is

  NoOfClients          : constant Positive := 10;
  NoOfExistingResources : constant Positive := 5;
  NoOfExistingInstances : constant Positive := 2;

  type Resource_Ix      is      range 1..NoOfExistingResources;
  subtype Instances_Available is Natural range 0..NoOfExistingInstances;
  type Resources_Available is array (Resource_Ix'Range) of Instances_Available;

  protected Resource_Controller is
    entry Get_Resource (Resource_Ix);
    procedure Release_Resource (Ix : in Resource_Ix);
    procedure Decrement_NoOfActiveClients;
    entry Wait_For_Deadlock_Or_Termination (Deadlocked : out Boolean);
  private
    function Deadlock_Or_Termination return Boolean;
    Resources          : Resources_Available := (others => NoOfExistingInstances);
    NoOfActiveClients : Natural             := NoOfClients;
  end Resource_Controller;

  protected body Resource_Controller is

    entry Get_Resource (for Ix in Resource_Ix) when Resources (Ix) > 0 is

      begin
        Resources (Ix) := Resources (Ix) - 1;
      end Get_Resource;

    procedure Release_Resource (Ix : in Resource_Ix) is

      begin
        Resources (Ix) := Resources (Ix) + 1;
      end Release_Resource;

    procedure Decrement_NoOfActiveClients is

      begin
        NoOfActiveClients := NoOfActiveClients - 1;
      end Decrement_NoOfActiveClients;

    function Deadlock_Or_Termination return Boolean is

      Waiting_Clients : Natural := 0;
    begin
      for i in Resource_Ix'Range loop
        Waiting_Clients := Waiting_Clients + Get_Resource (i)'Count;
      end loop;
      return Waiting_Clients = NoOfActiveClients;
    end Deadlock_Or_Termination;

    entry Wait_For_Deadlock_Or_Termination (Deadlocked : out Boolean)
    when Deadlock_Or_Termination is

      begin
        Deadlocked := NoOfActiveClients > 0;
      end Wait_For_Deadlock_Or_Termination;

  end Resource_Controller;

```

(i) [2 marks] How many task-queues are implemented by the protected object `Resource_Controller`? List and describe them.

(ii) [4 marks] The protected object `Resource_Controller` offers (besides allocation and deallocation of resource instances) a simple deadlock detection feature. It is assumed that the initial `NoOfClients` which claim (will try to allocate) a resource is known and that every client which has released all its resources and does no longer claim any resources will call `Decrement_NoOfActiveClients`. Describe how the implemented deadlock detection mechanism works.

(iii) [2 marks] Is this deadlock detection mechanism universal or will it only detect certain forms of deadlocks? Detail your answer in both cases precisely.

Now study the second part of the given Ada program carefully and read the questions on the next page.

```

task type Client;

task body Client is

    NoOfClaimedInstances : constant Positive := 1; -- case 1
-- NoOfClaimedInstances : constant Positive := 2; -- case 2

begin
    for Ix in Resource_Ix'Range loop
        for Instance in 1..NoOfClaimedInstances loop
            Resource_Controller.Get_Resource (Ix);
        end loop;
    end loop;

    for Ix in Resource_Ix'Range loop
        for Instance in 1..NoOfClaimedInstances loop
            Resource_Controller.Release_Resource (Ix);
        end loop;
    end loop;

    Resource_Controller.Decrement_NoOfActiveClients;
end Client;

Clients      : array (1..NoOfClients) of Client;
Deadlocked   : Boolean;

begin
    Resource_Controller.Wait_For_Deadlock_Or_Termination (Deadlocked);
    if Deadlocked then
        Put_Line("--- Deadlock detected, aborting clients ---");
        for i in Clients'Range loop
            abort Clients (i);
        end loop;
    else
        Put_Line("--- All clients terminated normally ---");
    end if;
end Synced_Processes;

```

(iv) [8 marks] Notice that 'case 1' is currently implemented ('case 2' is currently commented out and will be discussed in the next question). Will the program in the present 'case 1' behave deterministically and will it terminate, deadlock, or livelock? Which terminal output is to be expected? Give precise reasons for all your answers. If the program is found to be non-deterministic, discuss all possible outcomes.

(v) [6 marks] Now consider 'case 2' (imagine the corresponding line un-commented and the 'case 1' line commented out). Will the program now behave deterministically and will it terminate, deadlock, or livelock? Which terminal output is to be expected? Give precise reasons for all your answers. If the program is found to be non-deterministic, discuss all possible outcomes.

6. [38 marks] Distributed Systems

- (a) [6 marks] Exactly how does a requeue statement differ from an entry-call statement? Start answering the question by first enumerating in which regards they are similar or identical. Give an example where a requeue statement is required.

(b) [8 marks] Assume that you need to debug a large, distributed system.

(i) [4 marks] Why is it usually impossible to provide a global snapshot (i.e. a collection of all internal states) of the whole distributed system at any absolute time? Provide at least two precise reasons.

(ii) [4 marks] If it is not possible to provide a global snapshot of the whole distributed system at an absolute time, what qualifies a valid global snapshot instead? Give a precise definition.

(c) [20 marks] This question discusses several characteristics of transactions.

(i) [4 marks] Define serializability of two transactions and provide a test (or property) which guarantees serializability.

(ii) [6 marks] Discuss the individual ACID properties in the context of a distributed system. Specifically: which properties can become hard to fulfil in large scale distributed systems? Given precise reasons.

(iii) [4 marks] Can transactions be nested (called from within another transaction)? Give precise reasons or constraints to be considered.

(iv) [6 marks] In the case of time stamp ordering transaction schedulers, discuss the advantages and disadvantages of strict time stamp ordering.

(d) [4 marks] In terms of the OSI 7 layer model: how can a network router be defined and in which way is it different from a firewall?

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

continuation of answer to question part

continuation of answer to question part