

The Australian National University  
Mid Semester Examination – September 2015

## Comp2310 & Comp6310 Concurrent and Distributed Systems

Study period: 15 minutes  
Time allowed: 1.5 hours (after study period)  
Total marks: 50  
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.


*Student number:*

The following are for use by the examiners

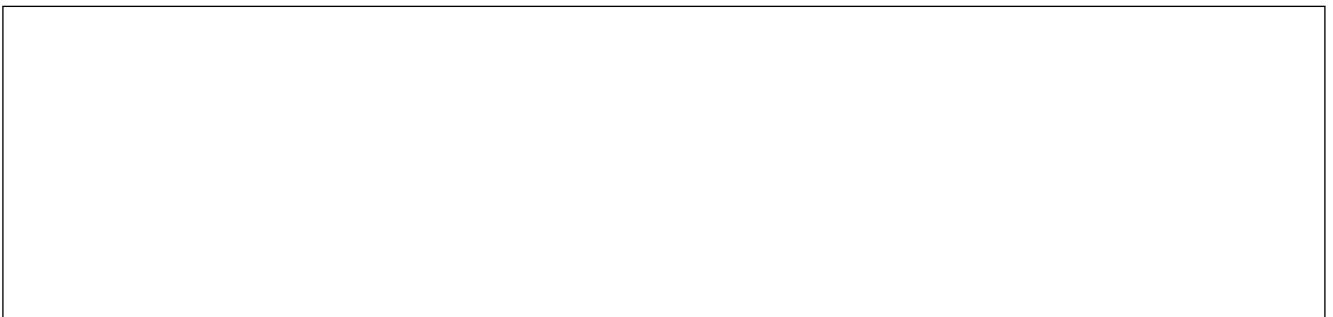
<i>Q1 mark</i>	<i>Q2 mark</i>	<i>Q3 mark</i>	<i>Total mark</i>

**1. [8 marks] General concurrency**

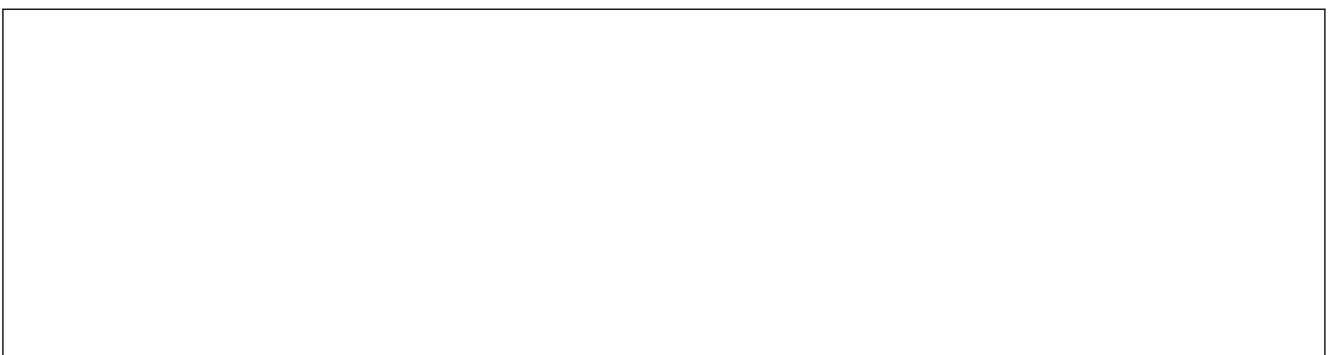
- (a) [4 marks] Draw a diagram showing all basic states of a process and all valid transitions between those states. Label all states with their names and all transitions with the events which cause those transitions.



- (b) [2 marks] The scheduler is executing the state transitions in your previous answer and thus controls which tasks are being executed on the CPU(s). But where and how is the scheduler itself being executed?



- (c) [2 marks] Assume that a concurrent program is executed on a single CPU (assume a single, sequential chain of machine instructions). Are mutual exclusion programming methods still required if accessing shared data? Or will the single CPU hardware enforce the correct program behavior anyway? Give precise reasons.



**2. [25 marks] Communication & Synchronization**

- (a) [8 marks] A number of  $n$  tasks all print out “Hello” followed sometime later by a “See you later”. Provide a program (in any programming language which you prefer, including pseudocode) to guarantee that no “See you later” will ever be issued before a “Hello” from any other task.

(b) [17 marks] Read the following Ada program carefully. The program is syntactically correct and will compile without warnings. See questions on the following pages.

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Postie is

  type Colours is (Red, Green, Blue);

  task Postbox is
    entry Set (C : Colours);
    entry Get (C : out Colours);
  end Postbox;

  task body Postbox is

    Store : Colours := Colours'Invalid_Value;

  begin
    loop
      select
        accept Set (C : Colours) do
          Store := C;
        end Set;
      or
        when Store'Valid =>
          accept Get (C : out Colours) do
            C := Store;
          end Get;
      or
        terminate;
      end select;
    end loop;
  end Postbox;

  protected Hand_Over is
    entry Give (C : Colours);
    entry Take (C : out Colours);
  private
    Store : Colours := Colours'Invalid_Value;
  end Hand_Over;

  protected body Hand_Over is

    entry Give (C : Colours)
      when not Store'Valid is
    begin
      Store := C;
    end Give;

    entry Take (C : out Colours)
      when Store'Valid is
    begin
      C := Store;
      Store := Colours'Invalid_Value;
    end Take;

  end Hand_Over;

  task type Actor (Init : Colours);
  task body Actor is

    Colour : Colours := Init;

  begin
    Postbox.Set (Colour);
    Hand_Over.Give (Colour);

    Hand_Over.Take (Colour);
    Put_Line ("Actor " & Colours'Image (Init)
      & " took " & Colours'Image (Colour));

    Postbox.Get (Colour);
    Put_Line ("Actor " & Colours'Image (Init)
      & " got " & Colours'Image (Colour));
  end Actor;

  Actor_Red : Actor (Red);
  Actor_Green : Actor (Green);
  Actor_Blue : Actor (Blue);

begin
  null;
end Postie;

```

Ada standard attributes used in this code:

The attribute `Store'Valid` is a language-defined, boolean function which returns `True` if and only if the value which is held by `Store` is a valid value of the type `Colours`.

The constant `Colours'Invalid_Value` provides a value which is invalid for the type `Colours`.

The function `Colours'Image` transforms a value of type `Colours` into a string representation of this value. For instance: `Colours'Image (Red)` returns `"RED"`.

(code continued in right column)

(i) [2 marks] How many tasks are implemented by this program? Name them.

(ii) [2 marks] How many task queues are implemented by this program? Name them.

(iii) [2 marks] A task could potentially be blocked at multiple operations inside this code. Enumerate those potentially blocking operations.

(iv) [2 marks] Is this program deterministic? Give precise reasons for your answer.

(v) [4 marks] Will this program terminate always, sometimes, or never? Give precise reasons for your answer.

(vi) [5 marks] What output (or multiple possible outputs) would you expect from running this program? Give precise reasons for your answer.

**3. [17 marks] Data-parallelism**

(a) [17 marks] Consider the function:

$$\sum_{k=1}^N (M - x_k)^2 = (M - x_1)^2 + (M - x_2)^2 + \dots + (M - x_N)^2$$

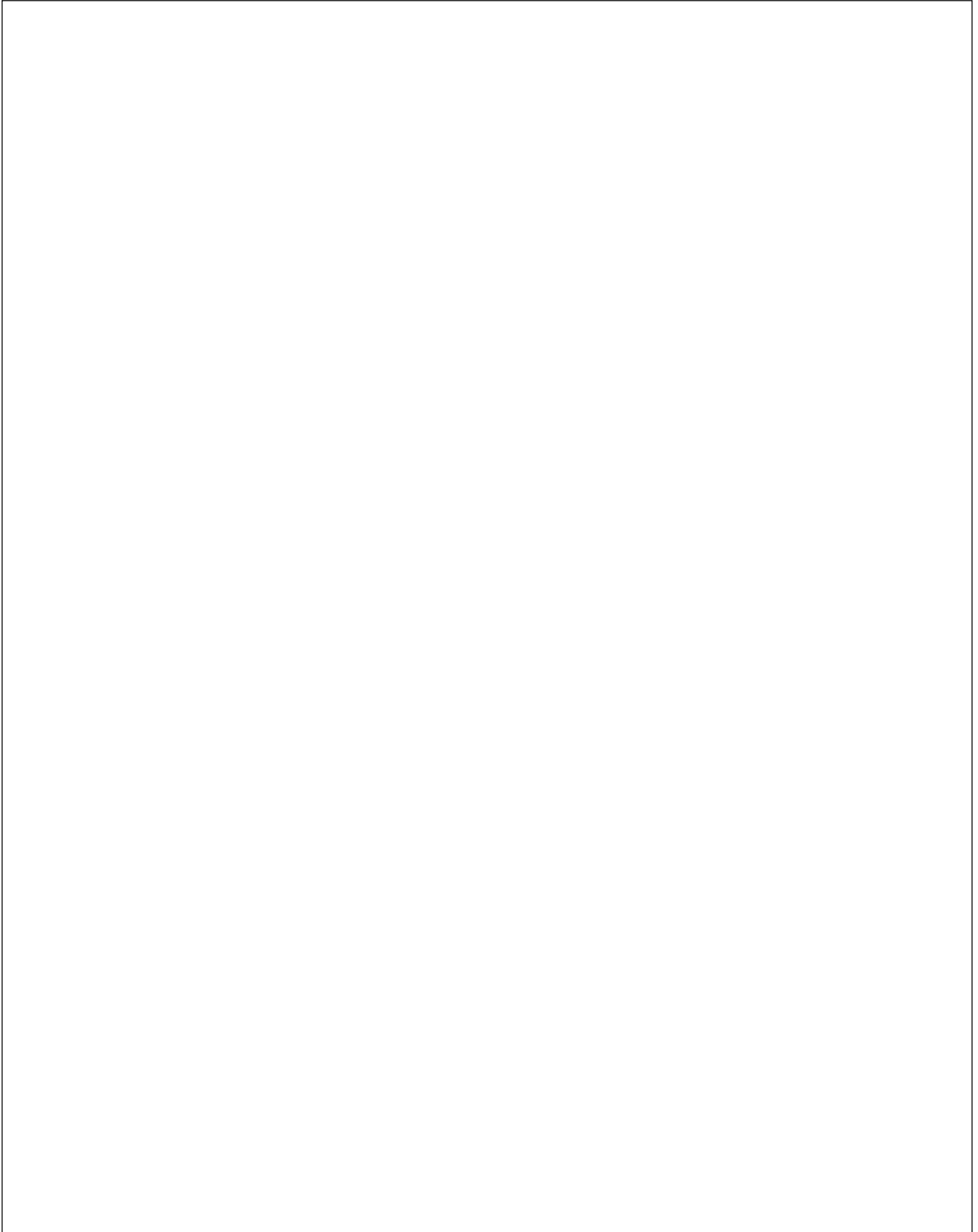
and then answer the following questions:

(i) [3 marks] Could a function like this be implemented with a concurrent program and gain performance compared to a sequential implementation? Give precise reasons.

(ii) [3 marks] How many concurrent tasks would you need for maximal performance? What would they do, i.e. which part of the function would be implemented by each task?

(iii) [3 marks] What is the computational time complexity of a sequential implementation? Will the computational complexity of the algorithm change for your concurrent implementation? If so: in what way? Give precise reasons for your answer either way.

(iv) [8 marks] Outline a concurrent implementation for the above function. We do not expect more than a sketch of a program from you here (do not spend all your exam time here to write a warning free program). You can use any programming language which you see fit, including pseudocode. If you have too much trouble expressing a basic structure in code, then draw a diagram outlining the stream of operations.





continuation of answer to question  part

continuation of answer to question  part

continuation of answer to question

part

continuation of answer to question

part