THE AUSTRALIAN NATIONAL UNIVERSITY

Mid Semester Examination, September 2013

COMP2310 / COMP6310
(Concurrent and Distributed Systems)

Writing Period: 1 hour duration

Study Period: 0 minutes duration

Permitted Materials: One A4 page with handwritten notes on both sides.
NO calculator permitted.

Questions are NOT equally weighted.
This exam will contribute 10% to your final assessment.

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.

The marking scheme will put a high value on clarity so, as a general guide, it is better to give fewer answers in a clear and concise manner than to outline a greater number in a sketchy, half-answered fashion.

Please write clearly – if we cannot read your writing you will lose marks!

Student Number (please write clearly):							

Enrollment (circle one):
COMP2310
COMP6310

Official use only:

Q1 (14)	Q2 (14)	Q3 (17)	Q4 (5)	Total (50)

QUESTION 1 [14 marks]

(a) Briefly describe the role of the run-time stack in the implementation of threads.

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

(b) In the implementation of Java threads, describe the relationship between the **`start()`** and **`run()`** methods.

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

(c) List the various ways to prevent deadlocks.

QUESTION 1(c) **[4 marks]**

(d) Define the terms *safety* and *liveness*. Briefly explain how these relate to the correctness of a concurrent program.

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

(e) In concurrent programming, there exist methodologies to go from requirements to model and from model to implementation. Why is it particularly important to apply such methodologies for constructing concurrent systems?

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

QUESTION 2 [14 marks]

(a) List all traces of length 2 for the FSP process:

```
SUM = (in[a:0..1][b:0..1] -> TOTAL[a+b]),
TOTAL[s:0..2] = (out[s] -> SUM).
```

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

(b) How many states does the process **SUM** have?

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

(c) A *binary semaphore*, in its initial state, can accept an **down** action, after which it can only accept a **up** action and returns to its initial state.

(i) Write an FSP model for the binary semaphore in the process **BSEM**.

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

(ii) An agent using a binary semaphore named **mutex** to guard the repeated **entry** and **exit** of a critical region may be modelled by:

```
AGENT = (mutex.down->enter->exit->mutex.up->AGENT).
```

Write an FSP parallel composition, **BSEMDEMO**, of a system with two agent processes and a binary semaphore such that only one agent can be in the critical region at any time.

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

(iii) Draw a Labelled Transition System (LTS) for the FSP property:

```
property MUTEX = (a[i:1..2].enter -> a[i].exit -> MUTEX).
```

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

(iv) **COMP2310 students**, *answer the following:* Specify an FSP composition of the system of Q2(c)(ii) to include the property `MUTEX` (noting that you may need to perform renaming of the action prefixes of `MUTEX`, depending on your previous answer). Would you expect the property to be violated? Briefly explain.

COMP6310 students, *answer the following:* For the system of Q2(c)(ii), specify an FSP *progress property* to express either thread will eventually be allowed to enter the critical region. Would you expect this property to be violated? Briefly explain. Specify a further FSP composition using *action priority* that could check this.

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

QUESTION 3 [17 marks]

(a) Describe the thread state transitions that are caused by calls to **`wait()`** and **`notify()`**. Be specific on which thread makes the call and which thread(s) makes the transition.

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

(b) In the context of concurrent programming, explain what is meant by the term *race condition*? Include in your answer 15 lines or less of pseudo code that has a race condition. *Hint:* the situation is sometimes also described as *interference between threads*.

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

Question 3 (continued)

(c) Consider the following Java code fragment of a monitor:

```
class AMonitor {
    private boolean aState = false; ...
    synchronized void accessMethod1() {
        while (aState == false) wait(); ...
    }
}
```

(i) Assuming that the monitor functioned as intended, what would be the effect of replacing the loop in `accessMethod1()` with a spin-loop, i.e. `while (aState == false) {/*spin*/}`

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

(ii) How could you implement the effect of the `synchronized` keyword (i.e. governing the entry and exit of a synchronized method) using a test-and-set operation? This operation is defined as follows:

`int testAndSet(volatile int *Lock):`
`{int lv = *Lock; *Lock = 1; return lv;}` (atomically)

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

(d) In the context of a concurrent program, what is a *monitor invariant*? Describe an illustrative example using pseudo-code.

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

Additional answers to QUESTION __(__)[__]

QUESTION 4 [5 marks]

(a) Describe a scenario where it would be useful to employ an Ada `select` statement (you need not describe it in terms of Ada).

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

(b) Give one advantage and one disadvantage of asynchronous message passing over synchronous message passing.

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

Additional answers to QUESTION ___(___)[___]

Additional answers to QUESTION ___(___)[___]

Additional answers to QUESTION ___(___)[___]