The University of New South Wales

Final Exam

2008/11/04

COMP3151/COMP9151

Foundations of Concurrency

Time allowed: **2 hours (8:45–11:00)**
Total number of questions: **5**
Total number of marks: **45**

Textbooks, lecture notes, etc. are not permitted, except for 2 double-sided A4 sheets of hand-written notes.

Calculators may not be used.

Not all questions are worth equal marks.

Answer all questions.

Answers must be written in ink.

You can answer the questions in any order.

You may take this question paper out of the exam.

Write your answers into the answer booklet provided. Be concise — *excessively verbose answers will be penalised*. Use a pencil or the back of the booklet for rough work. Your rough work will not be marked.

Shared-Variable Concurrency (15 Marks $\approx$ 40 minutes)

Recall that *starvation-freedom* is the liveness property relevant to mutual exclusion algorithms. It is satisfied if every process trying to enter its critical section will eventually do so.

Question 1 (10 marks)

Let A and B be two algorithms which were designed to solve the mutual exclusion problem, and let C be the algorithm obtained by replacing the critical section of A with the algorithm B:

Algorithm: C **(**n **processes)**
shared vars of A shared vars of B
loop forever
p1: non-critical section
p2: entry protocol of A
p3: entry protocol of B
p4: critical section
p5: exit protocol of B
p6: exit protocol of A

Assume that the shared variables of A are disjoint from those of B. Are the following statements correct? Justify each answer briefly (i.e., with a sentence or two).

(a) If both A and B are deadlock-free then C is deadlock-free.

(b) If both A and B are starvation-free then C is starvation-free.

(c) If either A or B satisfies mutual exclusion then C satisfies mutual exclusion.

(d) If A is deadlock-free and B is starvation-free then C is starvation-free.

(e) If A is starvation-free and B is deadlock-free then C is starvation-free.

Question 2 (5 marks)

Does the following mutual exclusion algorithm satisfy starvation-freedom? Sketch a proof or present a counter-example.

Algorithm: algorithm #3	
bit array b[0..1] $\leftarrow$ [0,0]	
p	**q**
loop forever	**loop forever**
p1: non-critical section	q1: non-critical section
p2: b[0] $\leftarrow$ 0	q2: b[1] $\leftarrow$ 1
p3: **while** b[0] $=$ 0 **do**	q3: **while** b[0] $=$ 1 **do**
p4: b[0] $\leftarrow$ 1	q4: b[1] $\leftarrow$ 1
p5: **while** b[1] $=$ 1 **do** b[0] $\leftarrow$ 0 **od**	q5: **while** b[1] $=$ 0 **do** b[1] $\leftarrow$ 0 **od**
od	**od**
p6: critical section	q6: critical section
p7: b[0] $\leftarrow$ 0	q7: b[1] $\leftarrow$ 0

Message-Passing Concurrency (30 Marks $\approx$ 80 minutes)

Question 3 (8 marks)

Hamming's problem. Use transition diagrams to present a message passing concurrent program $P = P_2 \parallel P_3 \parallel P_5 \parallel M$ whose output along channel Out is the sequence of all multiples of 2, 3, and 5 in strictly ascending order. The first elements of the sequence are 0, 2, 3, 4, 5, 6, 8, 9, 10, 12, 14. There will be four concurrent processes: one P_i each to calculate the multiples of the numbers $i = 2$, 3, and 5, respectively, and a fourth process M to merge the results.

Question 4 (12 marks)

Modify your solution P to Hamming's problem such that it terminates after k numbers have been sent to channel Out, where $k \in \mathbb{N}$ is a constant known to the merger process. (2 marks)

Define a post-condition ψ for P to capture the essential properties of P as specified above. (2 marks)

Outline a proof of $\{true\}\, P\, \{\psi\}$ (8 marks).

Question 5 (10 marks)

Recall the Ricart-Agrawala distributed mutual exclusion algorithm:

```
                 Algorithm: Ricart-Agrawala algorithm
                    integer myNum ← 0
                    set of node IDs deferred ← empty set
                    integer highestNum ← 0
                    boolean requestCS ← false
     Main
     loop forever
p1:     non-critical section
p2:     requestCS ← true
p3:     myNum ← highestNum + 1
p4:     for all other nodes N
p5:        send(request, N, myID, myNum)
p6:     await reply's from all other nodes
p7:     critical section
p8:     requestCS ← false
p9:     for all nodes N in deferred
p10:       remove N from deferred
p11:       send(reply, N, myID)
     Receive
     integer source, requestedNum
     loop forever
p12:    receive(request, source, requestedNum)
p13:    highestNum ← max(highestNum, requestedNum)
p14:    if not requestCS or requestedNum ≪ myNum
p15:       send(reply, source, myID)
p16:    else add source to deferred
```

(a) **4 marks:** Construct a scenario in which the ticket numbers are unbounded.

(b) **2 marks:** Can the deferred lists of all the nodes be non-empty?

(c) **2 marks:** What is the maximum number of entries in a single deferred list?

(d) **2 marks:** What is the maximum number of entries in all the deferred lists together?

Justify your answers to (b)–(d) briefly (i.e., with a sentence or two).