The University of New South Wales

Final Exam

14s2

COMP3151/COMP9151

Foundations of Concurrency

Time allowed: **2 hours**
Total number of questions: **4**
Total number of marks: **50**

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. Use a pencil or the back of the booklet for rough work. Your rough work will not be marked.

Shared-Variable Concurrency (20 Marks)

Question 1 (20 marks)

(a) 5 marks Recall Dekker's algorithm:

Dekker's algorithm	
boolean wantp ← false, wantq ← false integer turn ← 1	
p	**q**
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp ← true	q2: wantq ← true
p3: while wantq	q3: while wantp
p4: if turn = 2	q4: if turn = 1
p5: wantp ← false	q5: wantq ← false
p6: await turn = 1	q6: await turn = 2
p7: wantp ← true	q7: wantq ← true
p8: critical section	q8: critical section
p9: turn ← 2	q9: turn ← 1
p10: wantp ← false	q10: wantq ← false

It solves the critical section problem on sequentially consistent hardware. Prove by giving a counter example that Dekker's algorithm no longer guarantees mutual exclusion if we run it on x86-TSO.

(b) 10 marks Someone suggests the following adaptation to x86-TSO.

Dekker's algorithm TSO	
boolean wantp ← false, wantq ← false integer turn ← 1	
p	**q**
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp ← true	q2: wantq ← true
p3: MFENCE	q3: MFENCE
p4: while wantq	q4: while wantp
p5: if turn = 2	q5: if turn = 1
p6: wantp ← false	q6: wantq ← false
p7: MFENCE	q7: MFENCE
p8: await turn = 1	q8: await turn = 2
p9: wantp ← true	q9: wantq ← true
p10: MFENCE	q10: MFENCE
p11: critical section	q11: critical section
p12: turn ← 2	q12: turn ← 1
p13: wantp ← false	q13: wantq ← false
p14: MFENCE	q14: MFENCE

Prove that it guarantees mutual exclusion on x86-TSO.

(c) 5 marks Which MFENCE instructions can be omitted in pairs (to preserve symmetry) without sacrificing mutual exclusion? Justify your answer briefly.

Message-Passing Concurrency (30 Marks)

Answers to questions that require programming can be formulated using Ben-Ari's pseudo-code notation, Promela, or (if you must) C with MPI. Shared mutable state, semaphores, monitors etc. are not allowed in answers to the programming questions because this is the message passing section.

Question 2 (10 marks)

Given are n producers and a single consumer. They share an initially empty crate that can hold up to $k > 0$ products. The consumer naps until the crate is full, then consumes all products in the crate and returns to napping. Each producer repeatedly fabricates one product and puts it in the crate; the producer who fills the crate awakens the consumer. The crate is inaccessible for producers if it is full or if the consumer is in the process of comsuming.

Write a program to simulate the above with one process for the consumer and n processes for the producers.

Question 3 (8 marks)

Recall the Byzantine Generals problem.

> A group of Byzantine armies is surrounding an enemy city. The ballance of force is such that if they all attack together, they can capture the city; otherwise they must retreat in order to avoid defeat.
>
> The generals of the armies have reliable messengers who successfully deliver any message sent from one general to another. However, some of the generals may be *traitors* endeavouring to bring about defeat of the Byzantine armies by sending deceiving messages.

Under certain conditions the Byzantine General algorithm ensures that all loyal generals reach consensus on a plan and that this plan is almost the same as the majority vote of their initial choices; if the vote is tied, the final decision is to retreat.

(a) Suppose we have 7 generals, up to 2 of which might be traitors. For 4 marks, show why a third round of relaying messages is needed if the traitors do their best to confuse the loyal generals. That is, construct a scenario where two rounds do not suffice for the generals to make a decision.

(b) Drop one of the loyal generals and show for 4 marks that even a third or fourth round can't guarantee consensus for 6 generals with up to 2 traitors. Draw knowledge trees for two loyal generals that come to different conclusions.

Question 4 (12 marks)

Given two disjoint sets of integers S_0 and T_0, their union $S_0 \cup T_0$ has to be partitioned into two subsets S and T such that $|S| = |S_0|$ and $|T| = |T_0|$, and every element of S is smaller than every element of T.

Prove the synchronous transition diagram given below correct with respect to precondition

$$S = S_0 \wedge T = T_0 \wedge S \neq \emptyset \wedge S \cap T = \emptyset \wedge x \neq m \wedge y \neq n$$

and postcondition

$$S \cup T = S_0 \cup T_0 \wedge S \cap T = \emptyset \wedge |S| = |S_0| \wedge |T| = |T_0| \wedge \max S < \min T$$

where, by convention, $\min \emptyset = \infty$. Either (a) use Levin & Gries or AFR for full marks or (b) construct the product transition diagram and use Floyd for half marks.