THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester Examination, November 2012

COMP2310 / COMP6310
(Concurrent and Distributed Systems)

Writing Period: 3 hours duration

Study Period: 15 minutes duration

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

Questions are NOT equally weighted.
This exam will contribute 50% to your final assessment.
Write your answers in blue or black pen only. Answers written in pencil will not be marked.

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 manner than to outline a greater number in a sketchy, half-answered fashion.

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

Student Number (please write clearly):							

Enrollment (circle one):
COMP2310
COMP6310

Official use only:

Q1 (16)	Q2 (21)	Q3 (15)	Q4 (12)	Q5 (21)	Q6 (15)	Total (100)

QUESTION 1 [16 marks] General Concurrency

(a) A concurrent language must provide some concept of a task/thread or other concurrent entity. List two other elements that relate to concurrency and might be expected to be provided by a concurrent language.

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

(b) When several threads are created within a process, what areas of the process' memory can they share? Name an area which must be separate and briefly explain why.

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

(c) Give a definition of an *idempotent* operation. Provide two simple and distinct examples of idempotent operations in concurrent and distributed systems.

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

(d) What is the main similarity between the terms *concurrent system* and *distributed system*? What is the main difference (in terms of style of communication)?

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

(e) A function or subroutine can be classified as *thread-safe*. Give a definition of this term. Write a simple example of a function that is not thread-safe.

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

Additional answers to QUESTION 1 (__)[__]

Question 1 (continued)

(f) Two concurrent processes repeatedly engage in certain actions, some of which may be *shared*. Explain how labelled transition systems (LTS) may be used to model such processes and their concurrent composition. Include in your answer how synchronization is modelled, how deadlock can arise and how is time abstracted in this model.

QUESTION 1(f) **[5 marks]**

Additional answers to QUESTION 1 (__)[__]

QUESTION 2 [21 marks] Mutual Exclusion and Synchronization

(a) Consider the FSP description of a pre-paid gas station and its intended Java implementation given in the Appendix at the end of this booklet.

(i) The implementation has a number of problems (errors or deficiencies) that relate specifically to concurrent programming, with respect to being a correct implementation of the description and achieving its stated goals (see the Appendix).

For each such problem, state what it is, what its effect would be, and, if not obvious from your description, outline how you would correct it. Note that the effect of some problems do not become evident until others have been corrected.

QUESTION 2(a)[i] **[12 marks]**

Question 2 (continued)

QUESTION 2(a)[i] continued.

(ii) Describe appropriate defensive programming measures that should be added to the (corrected) implementation. Each class should not trust the other to behave correctly, according to the FSP description.

QUESTION 2(a)[ii] **[6 marks]**

(b) Can you (safely) call a monitor method from within a monitor method of a different object? Explain with a simple example.

QUESTION 2(b) **[3 marks]**

Additional answers to QUESTION 2 (__)[__]

QUESTION 3 [15 marks] Safety and Liveness

Consider the following algorithm for mutual exclusion for two threads, with `inCS0` and `inCS1` initially `false`.

```
while (true) { // thread 0
   // non-critical section 0
   ...
   while (inCS1==true) {/*spin*/}
   inCS0 = true;
   // critical section 0
   ...
   inCS0 = false;
}
```

```
while (true) { // thread 1
   // non-critical section 1
   ...
   while (inCS0==true){/*spin*/}
   inCS1 = true;
   // critical section 1
   ...
   inCS1 = false;
}
```

(a) Model the algorithm in FSP, capturing the essential characteristics for mutual exclusion.

Hint: an FSP process will model each thread. For each thread, model the execution of critical regions by the sequence **crit.enter** and **crit.exit**; that of the non-critical section as simply **noncrit**. The variables can be modelled as **inCS0:VAR** and **inCS1:VAR**, where

```
set VarAlpha = {read[0..1], write[0..1]}
VAR = VAR[0],
VAR[i:0..1] = ( read[i] -> VAR[i]
              | write[j:0..1] -> VAR[j] ).
```

Note also that the alphabets of the processes for each thread should be extended by {`inCS0, inCS1`}`.VarAlpha` before being composed into the final system.

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

(b) Does the algorithm ensure mutual exclusion? If so, briefly explain why. If not, state a sequence of actions (preferably in the form of a trace, the shorter the better) which shows how it is violated.

QUESTION 3(b) **[2 marks]**

(c) Specify an FSP safety property which could be used to formally check whether mutual exclusion is in fact preserved.

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

(d) Can the algorithm deadlock? Briefly explain why or why not.

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

Question 3 (continued)

(e) Specify FSP progress property(ies) which states that, once a thread is waiting to enter its critical region, it will eventually be allowed to enter. Briefly explain whether you expect it to be violated, i.e. whether starvation is possible.

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

Additional answers to QUESTION 3 (__)[__]

Student Number: ....

QUESTION 4 [12 marks] Message Passing

A bounded buffer can be specified in FSP as

```
BUFFER(N=5) = COUNT[0],
COUNT[ct:0..N] = ( when (ct < N) put -> COUNT[ct+1]
                 | when (ct > 0) get -> COUNT[ct-1]).
PRODUCER = ( put -> PRODUCER ).
CONSUMER = ( get -> CONSUMER ).
||BOUNDEDBUFFER = ( PRODUCER || BUFFER(5) || CONSUMER).
```

Write pseudo-code for an implementation of this system using synchronous message passing in which integer values are passed from the producer to the buffer, and from the buffer to the consumer. You may use the following classes:

```
public class Selectable {
    public void guard(boolean g);
}
public class Channel extends Selectable {
    public synchronized void send(int  v);
    public synchronized int receive();
}
public class Select {
    public void add(Selectable s);
    public synchronized int choose();
}
```

Note that you do not need to specify details of initialization and maintenance of the buffer's array, e.g. after receiving a value `v` from the producer, you can simply write something like `// insert item v into buffer` and specify how the buffer state variable `ct` should be changed. Include the code for the thread creation. As they are mostly similar, you need only give pseudo-code for one of the producer and the consumer, and then describe how the other differs from the first.

Note also that solutions using asynchronous message passing (i.e. the `Entry` class) will score 0 marks.

Hint: the consumer can send the buffer a (null) message when it is ready to consume an item; the buffer waits for an item to become available before receiving this, and then sends back the item.

QUESTION 4	**[12 marks]**

Question 4 (continued)

QUESTION 4 continued.

Student Number: ....

QUESTION 5 [21 marks] Architectures

(a) Three programming languages (paradigms) for concurrency are Ada95, Java and C/Posix. Briefly describe the characteristics of each in terms of level of abstraction, support for consistency (conversely the susceptibility to errors) and efficiency.

Illustrate this on a specific simple example such as a bounded buffer implementation.

QUESTION 5(a) **[6 marks]**

Question 5 (continued)

(b) Consider the following C program fragment:

```
int i, pids[4];
for (i=0; i < 4; i++)
  pids[i] = fork();
```

(i) Assuming there was one process before the loop was executed, how many processes would there be after the loop terminates?

QUESTION 5(b)[i] **[1 mark]**

(ii) After the loop, write pseudo-code that would determine whether this process is the original parent.

QUESTION 5(b)[ii] **[2 marks]**

(iii) Consider all processes created after the second iteration of the loop. Briefly explain how they will see consistent values of `pids[0]` and `pids[1]`. Suppose one of these then overwrites these values; will the other process see the effects of this? Briefly explain why.

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

(c) Can Unix pipes be applied to all possible communication scenarios in concurrent and distributed systems? If not, then what are their key limitations and what alternative method of Unix inter-process communication (IPC) would you use if these limitations were a problem?

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

(d) Write pseudo-code for a program which creates a child process which executes the command `./cmd` and then forces it to terminate after 10 seconds have elapsed. For full marks, it should only try to kill the child process if it is still running.

Hint: you may find the following system calls useful:
`int sleep(int seconds)`, `int kill(pid_t pid, int sig)`,
`int execl(const char *path, const char *arg, ...);`
`int sigaction(int signum, const struct sigaction *act,`
`struct sigaction *oldact);`, where `struct sigaction` {`void (*sa_handler)(int); ...`},
and **`SIGKILL`** and **`SIGCHLD`** are the symbolic names for the kill and terminated child signals. **`SIG_IGN`** is a value for the **`sa_handler`** field to ignore the signal.

QUESTION 5(d) **[6 marks]**

QUESTION 6 [15 marks] Distributed Systems

(a) What are the main advantages and disadvantages of the OSI model of network layers? What information must be included in messages to support this?

QUESTION 6(a) **[3 marks]**

(b) Briefly explain the term *router capacity*. For TCP/IP traffic, what happens if it is exceeded?

QUESTION 6(b) **[2 marks]**

(c) The User Datagram Protocol (UDP) is an unreliable protocol, whereas the Transport Control Protocol is deemed reliable. Why is UDP useful? Give an example in distributed systems where the UDP is preferred over TCP.

QUESTION 6(c) **[2 marks]**

(d) The three main methods of managing transactions are called two-phase locking, optimistic concurrency control, and timestamp ordering. Each imposes a *serialization* of all (committed) transactions. Briefly describe how is this achieved in each. What are the main strengths and weaknesses of the first two?

QUESTION 6(d) **[5 marks]**

Question 6 (continued)

(e) Briefly describe two ways an election algorithm is needed in order to deliver distributed transactions with replicated data. State under what circumstances would the algorithm be initiated.

QUESTION 6(e) **[3 marks]**

Additional answers to QUESTION __(__)[__]

Student Number: ....

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]

Additional answers to QUESTION __(__)[__]