Homework (Week 3)

Submission: Due on Friday, 24th of June, 11am Sydney Time. Please submit using the CSE Give System either online or via this command on a CSE terminal:

```
give cs3151 hw3 hw3.pdf
```

Please put all your answers in one PDF file called `hw3.pdf`. Use of LaTeX is encouraged but not required. Please make your answers as **concise as possible**.

Late submissions are accepted up to five days after the deadline, but at a penalty: 5% off your total mark per day.

Manna-Pnueli Algorithm (2 marks)

Recall the Manna-Pnueli algorithm from Lecture 3.

Manna-Pnueli Algorithm			
integer wantp, wantq $\leftarrow 0, 0$			
	forever do		**forever do**
p_1	*non-critical section*	q_1	*non-critical section*
p_2	**if** wantq $= -1$	q_2	**if** wantp $= -1$
	then wantp $\leftarrow -1$		**then** wantq $\leftarrow 1$
	else wantp $\leftarrow 1$		**else** wantq $\leftarrow -1$
p_3	**await** wantq $\neq$ wantp	q_3	**await** wantq $\neq -$wantp
p_4	*critical section*	q_4	*critical section*
p_5	wantp $\leftarrow 0$	q_5	wantq $\leftarrow 0$

Note: The p_2 and q_2 steps are one atomic step!

Recall that the if condition and body must be executed as one atomic step. If this were **not** the case, find an interleaving that violates mutual exclusion. That is, split the if condition into two steps (condition and body) and find an execution such that both processes end up in their critical section simultaneously.

Szymanski's Algorithm (5 marks)

In this Promela code archive you will find a Promela model of Szymanski's algorithm for three processes, broken down to satisfy LCR, with a particular choice of test order for the various **await** statements. This choice happens to satisfy mutual exclusion and eventual entry (as you may check in Spin), but as mentioned in the lectures, not all choices do.

The task here is to twiddle with the test orders and figure out which orderings break the algorithm and which don't. You don't need to test all permutations, but do answer these questions:

Can you find any reorderings that break mutual exclusion and/or eventual entry? (You should be able to find at least one). Are there any **await**s that don't seem sensitive to reordering at all? What if you reorder the tests for all the **await**s in the exact same way? And finally, based on any error trails you obtain, can you form an educated guess about **why** the test order matters?

Explain your findings, informally and in your own words.

Composing Solutions (5 Marks)

Let A and B be two algorithms purported to solve the mutual exclusion problem for two processes. 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 your answers with 1-2 sentences.

a) If either A or B satisfies mutual exclusion, then C satisfies mutual exclusion.

b) If A has no unnecessary delay and B satisfies mutual exclusion then C has no unnecessary delay.

c) If A satifies mutual exclusion and B has no unnecessary delay then C has no unnecessary delay.

d) If A is deadlock free and B guarantees eventual entry then C guarantees eventual entry.