7 Concurrent and Distributed Systems (RNW)

(*a*) (*i*) Name the four necessary conditions for deadlock. [2 marks]

(*ii*) Which of these conditions is frequently precluded in operating-system kernel designs in order to prevent deadlocks and why? [2 marks]

(*b*) Deadlocks are not limited to locks; cycles of waiting on *condition variables* can also lead to the "deadly embrace".

(*i*) Explain why it might be more difficult to debug deadlocks involving condition variables than those simply involving locks. [2 marks]

(*ii*) Briefly describe a condition-variable API change that might allow this problem to be solved in some cases; explain why it cannot always help. [2 marks]

(*c*) FreeBSD's WITNESS feature checks statically defined and dynamically discovered lock orders. Each time a lock is acquired, any previously undiscovered graph edges involving lock types currently held by the thread and the newly acquired lock type will be added to the graph. Cycle detection is performed, and debug information is printed if a previously unreported cycle is discovered.

(*i*) Describe a common code structure in which programmers are likely to be able to define a static order between two lock types. [2 marks]

(*ii*) Describe a common case in which programmers are likely to rely instead on dynamic discovery of an order between two lock types. [2 marks]

(*iii*) Unlike the deadlock-detection algorithm presented in lecture, the WITNESS algorithm does not remove edges when locks are released. Explain why WITNESS's behaviour might be more useful in practice. [2 marks]

(*iv*) WITNESS is subject to *false positives*: warnings can be emitted due to legitimate cycles even though, by design, the cycle could never trigger an actual deadlock. Describe a situation in which this might arise, and explain why deadlock could never occur. [3 marks]

(*v*) WITNESS, as written, is intended to be used with *mutexes* and other lock types providing mutual exclusion. A developer might naïvely extend WITNESS to support reader-writer locks (**rwlock**) by introducing graph edges for both read and write acquires as it does for mutex acquires. Explain why this might not always lead to the desired result. [3 marks]