8 Concurrent and Distributed Systems (RNW)

History graphs record dependencies between individual atomic operations within sequences of events associated with specific schedules of more complex *transactions*.

(*a*) (*i*) What do *edges* in a history graph represent? [1 mark]

(*ii*) What graph property holds if a *bad schedule* is present? [1 mark]

(*iii*) Which ACID properties may be violated by a bad schedule? [2 marks]

(*iv*) Define *serial* and *serialisable* executions. Explain whether (and if so, how) one is a superset of the other. [3 marks]

(*b*) Two transactions, **T1** and **T2**, consist of operations on two objects, **A** and **B**:

```
T1: {
    a = A.getbalance();
    b = B.getbalance();
    return (a + b);
}
```

```
T2 (v): {
    A.debit(v);
    B.credit(v);
}
```

(*i*) Explain how a *dirty read* might be experienced through concurrent executions of **T1** and **T2**. [2 marks]

(*ii*) Draw and label a history graph illustrating this bad schedule. [2 marks]

(*c*) A programmer designs a transaction system that uses history graphs to detect bad schedules. After an operation is performed, and before its containing transaction is allowed to commit, the history graph is updated and a graph analysis is run. If a bad schedule is detected, affected transactions will be aborted and rolled back.

(*i*) Will this scheme always make progress? Explain your answer. [2 marks]

(*ii*) *Time Stamp Ordering (TSO)* will sometimes reject good schedules, which could lead to unnecessary transaction aborts. Does the scheme described here accept or reject more schedules than TSO? Explain why. [3 marks]

(*iii*) Explain one way in which this scheme may perform better than TSO. Explain one way in which it may perform worse. [4 marks]