2010 Paper 5 Question 4

Concurrent and Distributed Systems

In an application, processes may be identified as “readers” or “writers” of a certain data object. Multiple-reader, single-writer access to this object must be implemented, with priority for writers over readers. Readers execute procedures *startread* and *endread* before and after reading. Writers execute procedures *startwrite* and *endwrite* before and after writing one-at-a-time.

The following variables are used in an implementation of the algorithm:

ar	is the count of active readers
rr	is the count of reading readers
aw	is the count of active writers
ww	is the count of writing writers (who write one-at-a-time)

(*a*) In a semaphore implementation:

For mutual exclusion:
SemCountGuard is a semaphore under which the above counts are read and written.
SemWrite is for writers to wait on, in order to write one-at-a-time.
For condition synchronisation:
SemOKtoRead is for readers to wait until all writers have finished.
SemOKtoWrite is for writers to wait until currently reading readers have finished.

Discuss the following pseudocode for an attempted implementation of *startread*:

```
procedure startread ( )
wait(SemCountGuard);
ar := ar + 1;
if aw > 0 then wait(SemOKtoRead);
rr := rr + 1;
signal(SemCountGuard)
return
```

[6 marks]

(*b*) Using the above example, comment on the ease of monitor programming and implementation, compared with semaphore programming. Assume a monitor *ReadersWriters* defines condition variables *OKtoRead* and*OKtoWrite*.

[6 marks]

(*c*) Describe and comment on the Java approach to supporting mutual exclusion and condition synchronisation. [4 marks]

(*d*) Explain how active objects and guarded commands avoid some of the issues arising in the above programs. [4 marks]