int readercount; // initialize to 0; number of readers currently accessing resource

// all semaphores initialised to 1

semaphore resource; // controls access (read/write) to the resource

semaphore rmutex; // for syncing changes to shared variable readcount

semaphore serviceQueue; // FAIRNESS: preserves ordering of requests (signaling must be FIFO)

// Reader section

reader() {

<ENTRY Section>

serviceQueue.P(); // wait in line to be serviced

rmutex.P(); // request exclusive access to readcount

readcount++; // update count of active readers

if (readcount == 1) // if I am the first reader

resource.P(); // request resource access for readers (writers blocked)

serviceQueue.V(); // let next in line be serviced

rmutex.V(); // release access to readcount

<CRITICAL Section>

//reading is performed

<EXIT Section>

rmutex.P(); // request exclusive access to readcount

readcount--; // update count of active readers

if (readcount == 0) // if there are no readers left

resource.V(); // release resource access for all

rmutex.V(); // release access to readcount

}

// Writer section

writer() {

<ENTRY Section>

serviceQueue.P(); // wait in line to be serviced

resource.P(); // request exclusive access to resource

serviceQueue.V(); // let next in line be serviced

<CRITICAL Section>

// writing is performed

<EXIT Section>

resource.V(); // release resource access for next reader/writer

}