```
int nr := 0, nw := 0, dr := 0, dw := 0
sem mutex := 1, r := 0, w := 0
readEnter() {
    P(mutex);          // need mutual exclusion
    if (nw > 0) {          // writer is in, better block
        dr++;          // increment number of delayed readers
        V(mutex);          // release mutual exclusion
        P(r);          // block on read semaphore
    }
    nr++;          // going to go in, record this
    if (dr > 0) {          // other delayed readers, let them in
        dr−−;          // one fewer delayed reader
        V(r);          // actually let reader in
    }
    else V(mutex);          // no readers waiting, release mutex
}
readExit() {
    P(mutex);          // need mutual exclusion
    nr−−;          // one reader is out
    if (nr == 0 and dw > 0) {          // last reader and writer waiting, so let it in
        dw−−;          // one fewer delayed writer
        V(w);          // actually let writer in
    }
    else V(mutex);          // release mutual exclusion
}
writeEnter() {
    P(mutex);          // need mutual exclusion
    if (nr > 0 or nw > 0) {          // someone else is in, better block
        dw++;          // increment number of delayed writers
        V(mutex);          // release mutual exclusion
        P(w);          // block on write semaphore
    }
    nw++;          // going to go in, record this
    V(mutex);          // release mutual exclusion
}
writeExit() {
    P(mutex);          // need mutual exclusion
    nw−−;          // one fewer writer in
    if (dr > 0)          // reader waiting, let it in
        dr−−; V(r);          // decrement and let in reader
    else if (dw > 0)          // writer waiting, let it in
        dw−−; V(w);          // decrement and let writer in
    else V(mutex);          // release mutual exclusion
}
```