```
int readcount, writecount; //(initial value = 0)
semaphore rmutex, wmutex, readTry, resource; //(initial value = 1)

//READER
reader() {
<ENTRY Section>
  wait(readTry); //Indicate a reader is trying to enter
  wait(rmutex); //lock entry section to avoid race condition with other readers
  readcount++; //report yourself as a reader
  if (readcount == 1) //checks if you are first reader
    wait(resource); //if you are first reader, lock the resource
  signal(rmutex); //release entry section for other readers
  signal(readTry); //indicate you are done trying to access the resource

<CRITICAL Section>
//reading is performed

<EXIT Section>
  wait(rmutex); //reserve exit section - avoids race condition with readers
  readcount--; //indicate you're leaving
  if (readcount == 0) //checks if you are last reader leaving
    signal(resource); //if last, you must release the locked resource
  signal(rmutex); //release exit section for other readers
}

//WRITER
writer() {
<ENTRY Section>
  wait(wmutex); //reserve entry section for writers - avoids race conditions
  writecount++; //report yourself as a writer entering
  if (writecount == 1) //checks if you're first writer
    wait(readTry); //if you're first, then you must lock the readers out. Prevent them from trying to
enter CS
  signal(wmutex); //release entry section
  wait(resource); //reserve the resource for yourself - prevents other writers from
simultaneously editing the shared resource
<CRITICAL Section>
  //writing is performed
  signal(resource); //release file

<EXIT Section>
  wait(wmutex); //reserve exit section
  writecount--; //indicate you're leaving
  if (writecount == 0) //checks if you're the last writer
    signal(readTry); //if you're last writer, you must unlock the readers. Allows them to try enter
CS for reading
  signal(wmutex); //release exit section
}
```