Complie and Execute

Q1: `gcc A2Q1.c -o a2q1 -lpthread` and `./a2q1 30 60`
Q3: `gcc A2Q3.c -o a2q3 -lpthread` and `./a2q3 30 60`

Question 1

Measurement

```
yma67@teaching:~/ws/ece427/a2$ gcc A2Q1.c -o a2q1 -lpthread
yma67@teaching:~/ws/ece427/a2$ ./a2q1 30 60
========[Reader Preference]=======
[Reader]>>>>>>>>>>>>>>>>>>>>>>>>>
[Max wait] 1.829000 ms
[Min wait] 0.003000 ms
[Avg wait] 0.111586 ms
[Count]    30000
[Writer]>>>>>>>>>>>>>>>>>>>>>>>>>
[Max wait] 3633.479000 ms
[Min wait] 0.003000 ms
[Avg wait] 120.850003 ms
[Count]    300
```

Observation

This implementation have a starvation problem that writer have to wait significantly longer than reader.

Reasoning

- All readers at a time slice should be considered to hold the `rw_mutex` lock if one of them actually holds, and therefore blocks all other pending writers
- Readers may give up rw_mutex lock only after `read_count` drops to 0, and a writer could write, or another reader could read, if they obtained the `rw_mutex` lock.
- Writer have to wait for readers if there are multiple readers, even if some readers arrives after the writer.
- For instance, if one reader is reading, and while the reader is reading, a writer followed by another n readers came, then the writers will wait for the (at least) `n+1` readers to finish their reading (even if n reader came after the writer), since the writer will not give up file mutex lock until `read_count` is 0.

Question 2

Starvation already observerd in Q1

Question 3

See `A2Q3.c`

CONTINUE to the next page

Question 4

Measurement

```
yma67@teaching:~/ws/ece427/a2$ gcc A2Q3.c -o a2q3 -lpthread
yma67@teaching:~/ws/ece427/a2$ ./a2q3 30 60
==============[Fair]==============
[Reader]>>>>>>>>>>>>>>>>>>>>>>>>>>
[Max wait] 408.932000 ms
[Min wait] 0.003000 ms
[Avg wait] 4.680607 ms
[Count]    30000
[Writer]>>>>>>>>>>>>>>>>>>>>>>>>>>
[Max wait] 374.021000 ms
[Min wait] 0.051000 ms
[Avg wait] 7.408260 ms
[Count]    300
```

Since `4.680607 ms` is not significantly higher than `7.408260 ms`, there is no starvation occurred.

Proof

```
std_dev_read  = range / 4 = 408.929 / 4 = 102.23225 (approx)
std_dev_write = range / 4 = 373.970 / 4 = 93.4925   (approx)
using Unpaired T-test, p-value is 0.6454, and even if alpha is 0.1, it is considered
as not significant.
```

END of this assignment