Peng Yun Kai

260864499

COMP 310 Assignment 2 Written Part

1. Initially, my program had no starvation issue. Both read and write would execute on average in an acceptable and similar amount of time. I believe that this is because there was no execution lasting long enough so that the reader would stay, and more readers could add on. Therefore, there wasn't really a constraint on the writer since the readers would execute too fast and the writers wouldn't actually starve.

```
Min write: 38.000000 ns
Min read: 49.000000 ns

Average write: 327.576667 ns
Average read: 422.151533 ns

Max write: 2208.000000 ns
Max read: 220066.000000 ns
```

2. I added an additional sleep parameter, usleep(2), inside of my read function so that it would simulate reading an actual input. With this added, the starvation issue shows up for the writer function so that the maximum waiting time of write as well as the average waiting time of write are ten folds larger than their respective times for reads.

```
Min write: 82.000000 ns
Min read: 44.000000 ns

Average write: 8171845.626667 ns
Average read: 1462.061100 ns

Max write: 148666146.000000 ns
Max read: 34672747.000000 ns
```

3. semaphore rw_mutex = 1;
   semaphore mutex = 1;
   sempahore in = 1
   int read_count = 0;

   //write function
   do {
   wait(in);
   wait(rw_mutex);
   ..
   // write is performed
   …
   signal(rw_mutex);
   signal(in);
   } while (true);

   //read function
   do {
   wait(in);
   wait(mutex);
   read_count++;
   if (read_count == 1)
   wait(rw_mutex);
   signal(mutex);
   signal(in);
   …
   // read is performed
   …
   wait(mutex);
   read_count--;
   if (read_count == 0)
   signal(rw_mutex);
   signal(mutex);
   } while (true);

4. My current implementation solves the reader writer problem such that there is no more starvation showing up in this version. Indeed, we can see that the maximum writing time and reading time are very similar.
   The implementation of an extra semaphore that is shared between them ensures that readers and writers have an equal chance of being processed such that there are no more writers waiting for an absurd amount of time after the readers.

```
Min write: 71.000000 ns
Min read: 60.000000 ns

Average write: 251867.040000 ns
Average read: 30720.517900 ns

Max write: 2393020.000000 ns
Max read: 4152581.000000 ns
```