First Reader-Writer Problem: No reader is kept waiting unless a writer has already obtained permission to use the shared object.

semaphore rw_mutex =1; (used as mutual exclusion for writers, also used by the first or last reader that enters or exits the critical section)

semaphore mutex=1; (used for ensure mutual exclusion, can be replaced by **Mutex (pthread_mutex_lock, pthread_mutex_unlock)**

int read_count=0;

Reader:

```
while (true) {
      wait(mutex);
      read_count++;
      if (read_count==1)  wait(rw_mutex);
      signal(mutex);

     /* Reading is performed */

      wait(mutex);
      read_count- -;
      if (read_count==0) signal(rw_mutex);
     signal(mutex);
}
```

Writer:

```
while(true) {
       wait(rw_mutex);

        /* Writing is performed*/

       signal(rw_mutex);

}
```

Second Reader-Writer Problem: No writer is kept waiting longer than absolutely necessary.

```
int readcount, writecount; (initial value = 0)
semaphore mutex_1, mutex_2, mutex_3, w, r ; (initial value = 1)
```

Reader:

```
While (true){
  wait(mutex_3);
    wait(r);
      wait(mutex_1);
        readcount := readcount + 1;
        if readcount == 1 then wait(w);
      signal(mutex_1);
    signal(r);
  signal(mutex_3);

//  reading is performed

  wait(mutex_1);
    readcount := readcount - 1;
    if readcount == 0 then signal(w);
  signal(mutex_1);
}
```

Writer:

```
while(true) {
  wait(mutex_2);
    writecount := writecount + 1;
    if writecount == 1 then wait(r);
  signal(mutex_2);

  wait(w);
     // writing is performed
  signal(w);

  wait(mutex_2);
    writecount := writecount - 1;
    if writecount == 0 then signal(r);
  signal(mutex_2);
}
```