```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <unistd.h>

#define NUM_READERS 5
#define NUM_WRITERS 2

sem_t mutex, write_block;
int data = 0, rcount = 0;

void *reader(void *arg) {
  int f = (int)arg;
  sem_wait(&mutex);
  rcount = rcount + 1;
  if (rcount == 1) {
    sem_wait(&write_block);
  }
  sem_post(&mutex);
  printf("Reader %d: read data as %d\n", f, data);
  sem_wait(&mutex);
  rcount = rcount - 1;
  if (rcount == 0) {
    sem_post(&write_block);
  }
  sem_post(&mutex);
}

void *writer(void *arg) {
  int f = (int)arg;
  sem_wait(&write_block);
```

```
  data++;
  printf("Writer %d: wrote data as %d\n", f, data);
  sem_post(&write_block);
}

int main() {
  pthread_t rtid[NUM_READERS], wtid[NUM_WRITERS];
  sem_init(&mutex, 0, 1);
  sem_init(&write_block, 0, 1);

  for (int i = 0; i < NUM_READERS; i++) {
    pthread_create(&rtid[i], NULL, reader, (void *)i);
  }
  for (int i = 0; i < NUM_WRITERS; i++) {
    pthread_create(&wtid[i], NULL, writer, (void *)i);
  }

  for (int i = 0; i < NUM_READERS; i++) {
    pthread_join(rtid[i], NULL);
  }
  for (int i = 0; i < NUM_WRITERS; i++) {
    pthread_join(wtid[i], NULL);
  }

  sem_destroy(&mutex);
  sem_destroy(&write_block);
  return 0;
```

kotlin

```
Reader 0: read data as 0
Reader 1: read data as 0
Reader 2: read data as 0
Writer 0: wrote data as 1
Reader 3: read data as 1
Reader 4: read data as 1
Writer 1: wrote data as 2
```

}