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

// Declare semaphores and shared variables
sem_t wrt;          // Controls writer access to the shared resource
pthread_mutex_t mutex; // Mutex to protect readcount variable
int readcount = 0;   // Number of readers currently reading
int data = 0;       // Shared resource (for demonstration)

// Function for readers
void* reader(void* arg) {
   int id = *(int*)arg;
   while (1) {
      // Entry section for readers
      pthread_mutex_lock(&mutex);
      readcount++;
      if (readcount == 1)
         sem_wait(&wrt); // First reader locks the writer
      pthread_mutex_unlock(&mutex);

      // Critical section for readers
      printf("Reader %d is reading data = %d\n", id, data);
      sleep(1);

      // Exit section for readers
      pthread_mutex_lock(&mutex);
      readcount--;
      if (readcount == 0)
         sem_post(&wrt); // Last reader unlocks the writer
```

```
        pthread_mutex_unlock(&mutex);

        sleep(1); // Simulate delay
    }
}

// Function for writers
void* writer(void* arg) {
    int id = *(int*)arg;
    while (1) {
        sem_wait(&wrt); // Wait for exclusive access

        // Critical section for writers
        data++;
        printf("\tWriter %d modified data to %d\n", id, data);
        sleep(2);

        sem_post(&wrt); // Release access

        sleep(2); // Simulate delay
    }
}

int main() {
    pthread_t rtid[5], wtid[3];
    int i, r[5], w[3];

    // Initialize semaphores and mutex
    sem_init(&wrt, 0, 1);
    pthread_mutex_init(&mutex, NULL);
```

```
    // Create 5 reader threads
    for (i = 0; i < 5; i++) {
        r[i] = i + 1;
        pthread_create(&rtid[i], NULL, reader, &r[i]);
    }

    // Create 3 writer threads
    for (i = 0; i < 3; i++) {
        w[i] = i + 1;
        pthread_create(&wtid[i], NULL, writer, &w[i]);
    }

    // Join threads (infinite loop demonstration)
    for (i = 0; i < 5; i++)
        pthread_join(rtid[i], NULL);
    for (i = 0; i < 3; i++)
        pthread_join(wtid[i], NULL);

    // Destroy semaphores and mutex
    sem_destroy(&wrt);
    pthread_mutex_destroy(&mutex);

    return 0;
}
```