Assignment 4b

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

int readcount = 0;       // number of active readers
int shared_data = 0;     // shared resource

sem_t x;     // binary semaphore to protect readcount
sem_t wsem;  // binary semaphore for readers/writers mutual
exclusion

void *reader(void *arg) {
    int id = *(int *)arg;

    while (1) {
        sem_wait(&x);         // lock readcount
        readcount++;
        if (readcount == 1)   // first reader locks wsem
            sem_wait(&wsem);
        sem_post(&x);         // unlock readcount

        // --- Critical Section (Reading) ---

        printf("Reader %d is reading shared_data = %d\n", id,
shared_data);
        sleep(1);

        sem_wait(&x);         // lock readcount
        readcount--;
        if (readcount == 0)   // last reader unlocks wsem
            sem_post(&wsem);
        sem_post(&x);         // unlock readcount

        sleep(1); // let others run
    }
    return NULL;
}

void *writer(void *arg) {
    int id = *(int *)arg;

    while (1) {
        sem_wait(&wsem);   // lock resource

        // --- Critical Section (Writing) ---
        shared_data++;
        printf("Writer %d is writing shared_data = %d\n", id,
shared_data);
        sleep(1);

        sem_post(&wsem);   // unlock resource
```

```
        sleep(2); // give readers chance
    }
    return NULL;
}

int main() {
    // pthread_t r[5], w[2];
    int num_readers = 5;
    int num_writers = 2;
    pthread_t *r = malloc(num_readers * sizeof(pthread_t));
    pthread_t *w = malloc(num_writers * sizeof(pthread_t));
    int r_ids[5] = {1, 2, 3, 4, 5};
    int w_ids[2] = {1, 2};

    // initialize semaphores
    sem_init(&x, 0, 1);
    sem_init(&wsem, 0, 1);

    // create threads
    for (int i = 0; i < 5; i++)
        pthread_create(&r[i], NULL, reader, &r_ids[i]);
    for (int i = 0; i < 2; i++)
        pthread_create(&w[i], NULL, writer, &w_ids[i]);

    // join threads (they loop infinitely)
    for (int i = 0; i < 5; i++)
        pthread_join(r[i], NULL);
    for (int i = 0; i < 2; i++)
        pthread_join(w[i], NULL);

    // cleanup (not normally reached)
    sem_destroy(&x);
    sem_destroy(&wsem);

    return 0;
}
```

Code:

```
manasvi@manasvi:/mnt/c/Users/bhute/Desktop/oslab/Ass 4$ ./four
Reader 1 is reading shared_data = 0
Reader 5 is reading shared_data = 0
Reader 3 is reading shared_data = 0
Reader 4 is reading shared_data = 0
Reader 2 is reading shared_data = 0
Writer 1 is writing shared_data = 1
Writer 2 is writing shared_data = 2
Reader 4 is reading shared_data = 2
Reader 2 is reading shared_data = 2
Reader 5 is reading shared_data = 2
Reader 3 is reading shared_data = 2
Reader 1 is reading shared_data = 2
Writer 1 is writing shared_data = 3
^C
```