Assignment 4 a

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

#define NO_OF_READERS 3
#define NO_OF_WRITERS 3

sem_t wsem;  // Semaphore for writer access control
pthread_mutex_t mutex; // Mutex for controlling access to readcount

int readcount = 0;  // Number of readers currently reading
int x = 0;  // Shared resource

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

    printf("Reader %d: Waiting to read\n", id);
    pthread_mutex_lock(&mutex);
    readcount++;
    if (readcount == 1) {
        sem_wait(&wsem);  // First reader blocks writers
    }
    pthread_mutex_unlock(&mutex);
```

```
    // Reading section
    printf("Reader %d: Reading x = %d\n", id, x);

    pthread_mutex_lock(&mutex);
    readcount--;
    if (readcount == 0) {
        sem_post(&wsem);  // Last reader allows writers
    }
    pthread_mutex_unlock(&mutex);

    printf("Reader %d: Finished reading\n", id);
    return NULL;
}

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

    printf("Writer %d: Waiting to write\n", id);
    sem_wait(&wsem);  // Wait for access to write

    // Writing section
    x++;  // Example operation
    printf("Writer %d: Writing x = %d\n", id, x);

    sem_post(&wsem);  // Signal that writing is done
```

```
    printf("Writer %d: Finished writing\n", id);
    return NULL;
}

int main() {
    pthread_t readers[NO_OF_READERS], writers[NO_OF_WRITERS];
    int reader_id[NO_OF_READERS], writer_id[NO_OF_WRITERS];

    // Initialize semaphore and mutex
    sem_init(&wsem, 0, 1);  // Binary semaphore for writer control
    pthread_mutex_init(&mutex, NULL);

    // Create reader threads
    for (int i = 0; i < NO_OF_READERS; i++) {
        reader_id[i] = i + 1;
        pthread_create(&readers[i], NULL, reader, &reader_id[i]);
    }

    // Create writer threads
    for (int i = 0; i < NO_OF_WRITERS; i++) {
        writer_id[i] = i + 1;
        pthread_create(&writers[i], NULL, writer, &writer_id[i]);
    }

    // Wait for all threads to complete
    for (int i = 0; i < NO_OF_READERS; i++) {
```

```
        pthread_join(readers[i], NULL);
    }

    for (int i = 0; i < NO_OF_WRITERS; i++) {
        pthread_join(writers[i], NULL);
    }

    // Destroy semaphore and mutex
    sem_destroy(&wsem);
    pthread_mutex_destroy(&mutex);

    return 0;
}
```

Example Output:

```
Writer 1: Waiting to write
Writer 1: Writing x = 1
Writer 1: Finished writing
Reader 1: Waiting to read
Reader 1: Reading x = 1
Reader 1: Finished reading
Writer 2: Waiting to write
Writer 2: Writing x = 2
Writer 2: Finished writing
Reader 2: Waiting to read
Reader 2: Reading x = 2
```

Reader 2: Finished reading

Writer 3: Waiting to write

Writer 3: Writing x = 3

Writer 3: Finished writing

Reader 3: Waiting to read

Reader 3: Reading x = 3

Reader 3: Finished reading