LAB 05

EXERCISE:

QUESTION 01: Implement the above code and paste the screen shot of the output.

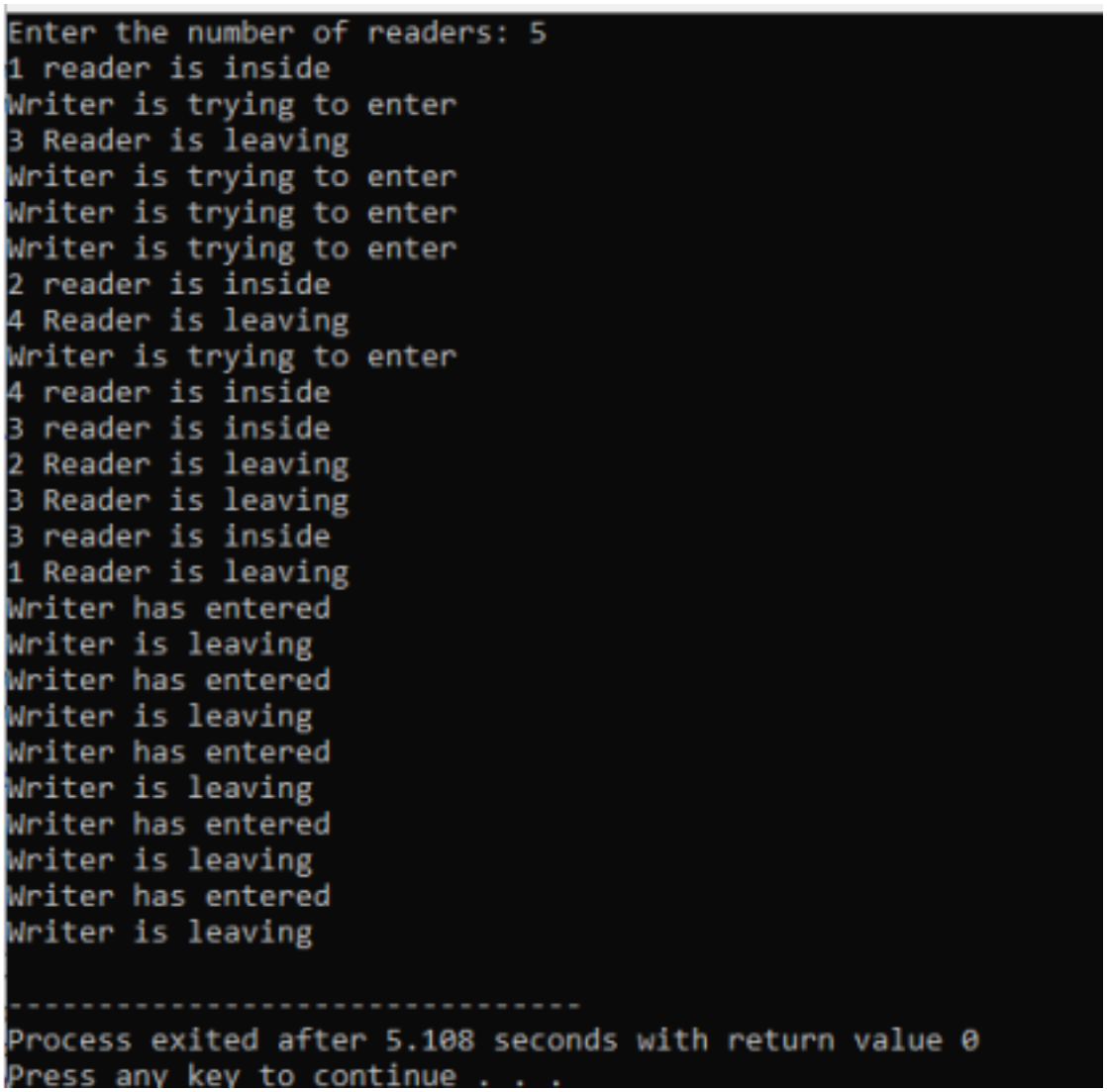
ANSWER:

CODE:

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

sem_t x, y; // Semaphores for synchronization
pthread_t writerthreads[100], readerthreads[100]; // Arrays for threads int
readercount = 0; // Number of readers currently in the critical section

// Reader thread function
void *reader(void *param)
{
    sem_wait(&x); // Enter the critical section for readercount manipulation
    readercount++;

    if (readercount == 1)
        sem_wait(&y); // The first reader blocks the writer

    sem_post(&x); // Exit the critical section for readercount manipulation

    printf("%d reader is inside\n", readercount);
    usleep(3); // Simulate reading (sleep)

    sem_wait(&x); // Enter critical section to manipulate readercount
    readercount--;

    if (readercount == 0)
        sem_post(&y); // Last reader releases the writer

    sem_post(&x); // Exit critical section for readercount manipulation

    printf("%d Reader is leaving\n", readercount + 1);
    return NULL;
}
```

```
// Writer thread function
void *writer(void *param)


{
    printf("Writer is trying to enter\n");
    sem_wait(&y); // Wait for the writer's turn
    printf("Writer has entered\n");
    sem_post(&y); // Allow other writers to enter

    printf("Writer is leaving\n");
    return NULL;
}

int main()
{
    int n2, i;

    // Get the number of readers
    printf("Enter the number of readers: ");
    scanf("%d", &n2);

    // Initialize semaphores
    sem_init(&x, 0, 1); // Semaphore for reader count
    sem_init(&y, 0, 1); // Semaphore for controlling writer access

    // Create reader and writer threads
    for (i = 0; i < n2; i++)
    {
        pthread_create(&readerthreads[i], NULL, reader, NULL);
        pthread_create(&writerthreads[i], NULL, writer, NULL);
    }

    // Wait for all threads to finish
    for (i = 0; i < n2; i++)
    {
        pthread_join(readerthreads[i], NULL);
        pthread_join(writerthreads[i], NULL);
    }

    // Destroy semaphores
    sem_destroy(&x);
    sem_destroy(&y);

    return 0;
}
```

OUTPUT:


Enter the number of readers: 5
1 reader is inside
Writer is trying to enter
3 Reader is leaving
Writer is trying to enter
Writer is trying to enter
Writer is trying to enter
2 reader is inside
4 Reader is leaving
Writer is trying to enter
4 reader is inside
3 reader is inside
2 Reader is leaving
3 Reader is leaving
3 reader is inside
1 Reader is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving
Writer has entered
Writer is leaving

Process exited after 5.108 seconds with return value 0
Press any key to continue . . .