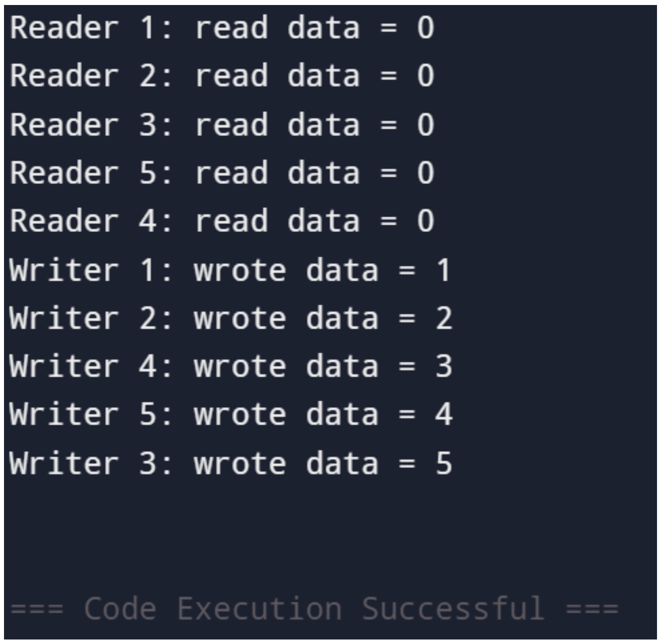
EXP 20: Construct a C program to simulate Reader-Writer problem using Semaphores

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


sem_t mutex, writeblock;

int data = 0;       // Shared data

int readcount = 0;   // Number of readers


void* reader(void* arg) {

  int f = *((int*)arg);

  sem_wait(&mutex);

  readcount++;

  if (readcount == 1)

    sem_wait(&writeblock); // First reader locks writers

  sem_post(&mutex);


  // Reading section

  printf("Reader %d: read data = %d\n", f, data);

  sleep(1);


  sem_wait(&mutex);

  readcount--;

  if (readcount == 0)
```

```
        sem_post(&writeblock); // Last reader unlocks writers
    sem_post(&mutex);
    return NULL;
}

void* writer(void* arg) {
    int f = *((int*)arg);
    sem_wait(&writeblock); // Only one writer at a time

    // Writing section
    data++;
    printf("Writer %d: wrote data = %d\n", f, data);
    sleep(1);

    sem_post(&writeblock);
    return NULL;
}

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

    sem_init(&mutex, 0, 1);
    sem_init(&writeblock, 0, 1);

    int reader_ids[5] = {1, 2, 3, 4, 5};
    int writer_ids[5] = {1, 2, 3, 4, 5};
    // Create reader and writer threads
    for (i = 0; i < 5; i++) {
        pthread_create(&rtid[i], NULL, reader, &reader_ids[i]);
        pthread_create(&wtid[i], NULL, writer, &writer_ids[i]);
    }

    // Wait for all threads
    for (i = 0; i < 5; i++) {
        pthread_join(rtid[i], NULL);
        pthread_join(wtid[i], NULL);
    }

    // Cleanup
    sem_destroy(&mutex);
    sem_destroy(&writeblock);

    return 0;
}
```

Sample Output



```
Reader 1: read data = 0
Reader 2: read data = 0
Reader 3: read data = 0
Reader 5: read data = 0
Reader 4: read data = 0
Writer 1: wrote data = 1
Writer 2: wrote data = 2
Writer 4: wrote data = 3
Writer 5: wrote data = 4
Writer 3: wrote data = 5


=== Code Execution Successful ===
```