Experiment Number: 6

Problem Statement: **a) : Implementation of Classical problem  Reader-writer using Threads and Semaphore**

**b)Implementation of Classical problem Reader-writer  using Threads and Mutex**

NAME: Aadesh Chawla ROLLNO: 12

CLASS: TY-IT-A BATCH: B1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Using Semaphore:**

#include <iostream>

#include <thread>

#include <semaphore.h>

#include <chrono>

using namespace std;

const int NUM\_READERS = 2;

const int NUM\_WRITERS = 1;

sem\_t rw\_mutex;

sem\_t mutex;

int readers\_count = 0;

void Reader(int id) {

    while (true) {

        this\_thread::sleep\_for(chrono::milliseconds(1000));

        sem\_wait(&mutex);

        readers\_count++;

        if (readers\_count == 1) {

            sem\_wait(&rw\_mutex);

        }

        sem\_post(&mutex);

        cout << "Reader " << id << " enters the critical section. Total readers: " << readers\_count << endl;

        sem\_wait(&mutex);

        readers\_count--;

        if (readers\_count == 0) {

            sem\_post(&rw\_mutex);

        }

        sem\_post(&mutex);

        // Simulate reading

        this\_thread::sleep\_for(chrono::milliseconds(2000));

        cout << "Reader " << id << " leaves the critical section. Total readers: " << readers\_count << endl;

    }

}

void Writer(int id) {

    while (true) {

        this\_thread::sleep\_for(chrono::milliseconds(2000));

        sem\_wait(&rw\_mutex);

        cout << "Writer " << id << " enters the critical section." << endl;

        this\_thread::sleep\_for(chrono::milliseconds(3000));

        cout << "Writer " << id << " leaves the critical section." << endl;

        sem\_post(&rw\_mutex);

    }

}

int main() {

    sem\_init(&rw\_mutex, 0, 1);

    sem\_init(&mutex, 0, 1);

    thread readers[NUM\_READERS];

    thread writers[NUM\_WRITERS];

    for (int i = 0; i < NUM\_READERS; i++) {

        readers[i] = thread(Reader, i + 1);

    }

    for (int i = 0; i < NUM\_WRITERS; i++) {

        writers[i] = thread(Writer, i + 1);

    }

    for (int i = 0; i < NUM\_READERS; i++) {

        readers[i].join();

    }

    for (int i = 0; i < NUM\_WRITERS; i++) {

        writers[i].join();

    }

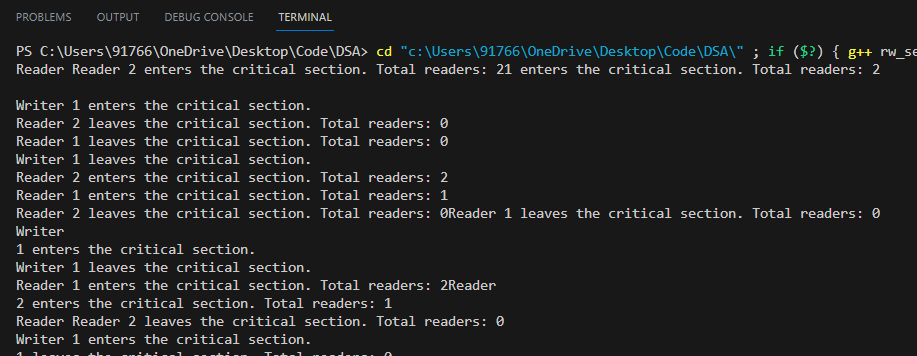
    sem\_destroy(&rw\_mutex);

    sem\_destroy(&mutex);

    return 0;

}

**Output:**



**Using Mutex:**

#include <iostream>

#include <thread>

#include <mutex>

#include <chrono>

using namespace std;

const int NUM\_READERS = 2;

const int NUM\_WRITERS = 1;

mutex mtx;

int readers\_count = 0;

void Reader(int id) {

    while (true) {

        this\_thread::sleep\_for(chrono::milliseconds(1000));

        mtx.lock();

        readers\_count++;

        if (readers\_count == 1) {

            cout << "Reader " << id << " enters the critical section. Total readers: " << readers\_count << endl;

        }

        mtx.unlock();

        this\_thread::sleep\_for(chrono::milliseconds(2000));

        mtx.lock();

        readers\_count--;

        if (readers\_count == 0) {

            cout << "Reader " << id << " leaves the critical section. Total readers: " << readers\_count << endl;

        }

        mtx.unlock();

    }

}

void Writer(int id) {

    while (true) {

        this\_thread::sleep\_for(chrono::milliseconds(2000));

        cout << "Writer " << id << " requests to write." << endl;

        mtx.lock();

        cout << "Writer " << id << " enters the critical section." << endl;

        mtx.unlock();

        this\_thread::sleep\_for(chrono::milliseconds(3000));

        mtx.lock();

        cout << "Writer " << id << " leaves the critical section." << endl;

        mtx.unlock();

    }

}

int main() {

    thread readers[NUM\_READERS];

    thread writers[NUM\_WRITERS];

    for (int i = 0; i < NUM\_READERS; i++) {

        readers[i] = thread(Reader, i + 1);

    }

    for (int i = 0; i < NUM\_WRITERS; i++) {

        writers[i] = thread(Writer, i + 1);

    }

    for (int i = 0; i < NUM\_READERS; i++) {

        readers[i].join();

    }

    for (int i = 0; i < NUM\_WRITERS; i++) {

        writers[i].join();

    }

    return 0;

}

**Output:**

