ASSIGNMENT 4

PROGRAM

**With Synchronization:**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <mutex>

#include <condition\_variable>

using namespace std;

sem\_t wrt; // Semaphore for writer

mutex mtx; // Mutex for reader count synchronization

int cnt = 1; // Shared resource

int numreader = 0; // Reader count

void\* writer(void\* wno)

{

sem\_wait(&wrt); // Wait for the semaphore, block if a writer is currently active

cnt = cnt \* 2; // Writer modifies shared resource

printf("Writer %d modified cnt to %d\n", \*((int\*)wno), cnt);

sem\_post(&wrt); // Signal that the writer has finished

return NULL;

}

void\* reader(void\* rno)

{

unique\_lock<mutex> lock(mtx); // Lock the mutex

numreader++; // Increment reader count

if (numreader == 1) {

sem\_wait(&wrt); // If it's the first reader, block the writer

}

lock.unlock(); // Release the lock to allow other readers to enter

// Reading Section

printf("Reader %d: read cnt as %d\n", \*((int\*)rno), cnt);

lock.lock(); // Reacquire the lock

numreader--; // Decrement reader count

if (numreader == 0) {

sem\_post(&wrt); // If it's the last reader, unblock the writer

}

lock.unlock(); // Release the lock

return NULL;

}

int main()

{

pthread\_t read[10], write[5];

sem\_init(&wrt, 0, 1); // Initialize the semaphore with value 1

int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

// Create reader threads

for (int i = 0; i < 10; i++) {

pthread\_create(&read[i], NULL, reader, (void\*)&a[i]);

}

// Create writer threads

for (int i = 0; i < 5; i++) {

pthread\_create(&write[i], NULL, writer, (void\*)&a[i]);

}

// Join reader threads

for (int i = 0; i < 10; i++) {

pthread\_join(read[i], NULL);

}

// Join writer threads

for (int i = 0; i < 5; i++) {

pthread\_join(write[i], NULL);

}

sem\_destroy(&wrt); // Destroy semaphore

return 0;

}

**Without Synchronization:**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

using namespace std;

sem\_t wrt; // Semaphore for writer

int cnt = 1; // Shared resource

int numreader = 0; // Reader count

void\* writer(void\* wno)

{

sem\_wait(&wrt); // Wait for the semaphore, block if a writer is currently active

cnt = cnt \* 2; // Writer modifies shared resource

printf("Writer %d modified cnt to %d\n", \*((int\*)wno), cnt);

sem\_post(&wrt); // Signal that the writer has finished

return NULL;

}

void\* reader(void\* rno)

{

numreader++; // Increment reader count

if (numreader == 1) {

sem\_wait(&wrt); // If it's the first reader, block the writer

}

// Reading Section

printf("Reader %d: read cnt as %d\n", \*((int\*)rno), cnt);

numreader--; // Decrement reader count

if (numreader == 0) {

sem\_post(&wrt); // If it's the last reader, unblock the writer

}

return NULL;

}

int main()

{

pthread\_t read[10], write[5];

sem\_init(&wrt, 0, 1); // Initialize the semaphore with value 1

int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

// Create reader threads

for (int i = 0; i < 10; i++) {

pthread\_create(&read[i], NULL, reader, (void\*)&a[i]);

}

// Create writer threads

for (int i = 0; i < 5; i++) {

pthread\_create(&write[i], NULL, writer, (void\*)&a[i]);

}

// Join reader threads

for (int i = 0; i < 10; i++) {

pthread\_join(read[i], NULL);

}

// Join writer threads

for (int i = 0; i < 5; i++) {

pthread\_join(write[i], NULL);

}

sem\_destroy(&wrt); // Destroy semaphore

return 0;

}

OUTPUT:

with sync

Reader 1: read cnt as 1

Reader 2: read cnt as 1

Reader 8: read cnt as 1

Writer 1 modified cnt to 2

Writer 4 modified cnt to 4

Reader 4: read cnt as 4

Reader 3: read cnt as 4

Reader 6: read cnt as 4

Reader 5: read cnt as 4

Reader 7: read cnt as 4

Reader 9: read cnt as 4

Reader 10: read cnt as 4

Writer 2 modified cnt to 8

Writer 3 modified cnt to 16

Writer 5 modified cnt to 32

=== Code Execution Successful ===

without sync

Reader 3: read cnt as 1

Reader 4: read cnt as 1

Reader 1: read cnt as 1

Reader 5: read cnt as 1

Reader 8: read cnt as 1

Reader 6: read cnt as 1

Reader 7: read cnt as 1

Reader 2: read cnt as 1

Reader 10: read cnt as 1

Writer 1 modified cnt to 2

Writer 2 modified cnt to 4

Writer 3 modified cnt to 8

Writer 4 modified cnt to 16

Writer 5 modified cnt to 32

Reader 9: read cnt as 32

=== Code Execution Successful ===