**EXPERIMENT – 29**

29. Write a C program to simulate the solution of Classical Process Synchronization Problem

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define SIZE 5

int buffer[SIZE];

int in = 0, out = 0;

sem\_t empty, full;

pthread\_mutex\_t mutex;

void\* producer(void\* arg) {

int item;

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

item = rand() % 100;

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

buffer[in] = item;

printf("Producer produced: %d\n", item);

in = (in + 1) % SIZE;

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

sleep(1);

}

return NULL;

}

void\* consumer(void\* arg) {

int item;

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

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

item = buffer[out];

printf("Consumer consumed: %d\n", item);

out = (out + 1) % SIZE;

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty

sleep(2);

}

return NULL;

}

int main() {

pthread\_t prod, cons;

sem\_init(&empty, 0, SIZE);

sem\_init(&full, 0, 0);

pthread\_mutex\_init(&mutex, NULL);

pthread\_create(&prod, NULL, producer, NULL);

pthread\_create(&cons, NULL, consumer, NULL);

pthread\_join(prod, NULL);

pthread\_join(cons, NULL);

sem\_destroy(&empty);

sem\_destroy(&full);

pthread\_mutex\_destroy(&mutex);

return 0;

}

SAMPLE OUTPUT:

Producer produced: 12

Consumer consumed: 12

Producer produced: 55

Producer produced: 34

Consumer consumed: 55