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

#include <stdio.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, mutex;

void\* producer(void\* arg) {

int item = 1;

while (1) {

sem\_wait(&empty);

sem\_wait(&mutex);

buffer[in] = item;  
 printf("Producer produced: %d\n", item);  
 in = (in + 1) % SIZE;  
 item++;  
  
 sem\_post(&mutex);  
 sem\_post(&full);  
  
 sleep(1);

}

}

void\* consumer(void\* arg) {

while (1) {

sem\_wait(&full);

sem\_wait(&mutex);

int item = buffer[out];  
 printf("Consumer consumed: %d\n", item);

out = (out + 1) % SIZE;

sem\_post(&mutex);

sem\_post(&empty);

sleep(2);

}

}

int main() {

pthread\_t prod, cons;

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

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

sem\_init(&mutex, 0, 1);

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);

sem\_destroy(&mutex);

return 0;

}