```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

// Producer-Consumer (no semaphores)

pthread_mutex_t mutex;
pthread_cond_t condFull;
pthread_cond_t condEmpty;
const int NUM_CONSUMER=5; // more than one!
const int NUM_PRODUCER=2; // more than one!
const int DIM_BUFFER=10;
int buffer[DIM_BUFFER];
int head=0;

void* produce(void* arg) {
    while(true){
    sleep(1);
    pthread_mutex_lock(&mutex);
    while(head>=DIM_BUFFER){
         pthread_cond_wait(&condFull,&mutex);
    }

    int num =rand()%100;
    buffer[head]=num;
    head++;
    printf("produce: %d\n",num);
    pthread_mutex_unlock(&mutex);
    pthread_cond_broadcast(&condEmpty);
    }
    return NULL;
}

void* consume(void* arg) {
    while(true){
    sleep(1);
    pthread_mutex_lock(&mutex);
    while(head==0){
         pthread_cond_wait(&condEmpty,&mutex);
    }

    int num = buffer[head-1];
    head--;
    printf("head: %d consume: %d\n",head,num);
    pthread_mutex_unlock(&mutex);
    pthread_cond_broadcast(&condFull);
    }
    return NULL;
}

int main(int argc, char* argv[]) {
    pthread_t thC[NUM_CONSUMER];
    pthread_t thP[NUM_PRODUCER];
    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&condFull, NULL);
    pthread_cond_init(&condEmpty, NULL);
    for (int i = 0; i < NUM_CONSUMER; i++) {
    if (pthread_create(&thC[i], NULL, &consume, NULL) != 0) {
        perror("Failed to create thread");
    }
    }
    for (int i = 0; i < NUM_PRODUCER; i++) {
    if (pthread_create(&thP[i], NULL, &produce, NULL) != 0) {
        perror("Failed to create thread");
    }
    }
    for (int i = 0; i < NUM_CONSUMER; i++) {
        if (pthread_join(thC[i], NULL) != 0) {
            perror("Failed to join thread");
        }
    }
    for (int i = 0; i < NUM_PRODUCER; i++) {
        if (pthread_join(thP[i], NULL) != 0) {
            perror("Failed to join thread");
        }
    }
    pthread_mutex_destroy(&mutex);
    pthread_cond_destroy(&condFull);
    pthread_cond_destroy(&condEmpty);
    return 0;
}

```