```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdlib.h>

#define BUFFER_SIZE 5

int buffer[BUFFER_SIZE];
int in = 0, out = 0, count = 0;

sem_t full, empty;
pthread_mutex_t mutex;

void *producer(void *arg) {
    int id = *(int*)arg;
    free(arg);  // Avoid memory leak from malloc
    while(1) {
        int item = rand() % 100;  // Produce a random item

        sem_wait(&empty);
        pthread_mutex_lock(&mutex);

        // Add item to buffer
        buffer[in] = item;
        printf("Producer %d produced: %d at position %d\n", id, item, in);
        in = (in + 1) % BUFFER_SIZE;
        count++;

        pthread_mutex_unlock(&mutex);
        sem_post(&full);

        sleep(1);  // Simulate production time
    }
    pthread_exit(NULL);
}

void *consumer(void *arg) {
    int id = *(int*)arg;
    free(arg);
    while(1) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);

        // Remove item from buffer
        int item = buffer[out];
        printf("Consumer %d consumed: %d from position %d\n", id, item, out);
        out = (out + 1) % BUFFER_SIZE;
        count--;

        pthread_mutex_unlock(&mutex);
        sem_post(&empty);

        sleep(1);  // Simulate consumption time
    }
    pthread_exit(NULL);
}

int main() {
    int p, c;
    pthread_t producers[10], consumers[10];

    pthread_mutex_init(&mutex, NULL);
    sem_init(&full, 0, 0);
    sem_init(&empty, 0, BUFFER_SIZE);

    printf("Enter number of producers: ");
    scanf("%d", &p);
    printf("Enter number of consumers: ");
    scanf("%d", &c);

    // Create producer threads
    for(int i = 0; i < p; i++) {
        int *id = malloc(sizeof(int));  // Each thread gets its own copy
        *id = i + 1;
        pthread_create(&producers[i], NULL, producer, id);
    }

    // Create consumer threads
    for(int i = 0; i < c; i++) {
        int *id = malloc(sizeof(int));
        *id = i + 1;
        pthread_create(&consumers[i], NULL, consumer, id);
    }

    // Wait for threads (infinite loops, so Ctrl+C to stop in real run)
    for(int i = 0; i < p; i++) {
        pthread_join(producers[i], NULL);
    }
    for(int i = 0; i < c; i++) {
        pthread_join(consumers[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&full);
    sem_destroy(&empty);
    return 0;
}

/*OUTPUT –
Enter number of producers: 2
Enter number of consumers: 2
Producer 1 produced: 52 at position 0
Producer 2 produced: 23 at position 1
Consumer 1 consumed: 52 from position 0
Consumer 2 consumed: 23 from position 1
Producer 1 produced: 77 at position 2
Consumer 1 consumed: 77 from position 2
Producer 2 produced: 99 at position 3
Consumer 2 consumed: 99 from position 3
...
*/
```