```
 PGMS=Producer_Consumer_1 Producer_Consumer_2 Producer_Consumer_3

CC=gcc
CFLAGS=-Wall -pthread

all: $(PGMS)

clean:
    rm -f $(PGMS)
    rm -f *.o
end
```

```c
// Empty/Full with no lock

// Note: All pthread and sem functions should have their return codes
//       checked. The checking has been omitted to clarity in
//       this example.

#include <assert.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <unistd.h>

#define MAX 3

int buffer[MAX];
int fill = 0;
int use = 0;
int loops = 20;

sem_t empty;
sem_t full;

void put(int value)
{
    buffer[fill] = value;
    fill = (fill + 1) % MAX;
}

int get()
{
    int tmp = buffer[use];
    use = (use + 1) % MAX;

    return tmp;
}

void *producer(void *arg)
{
    int val = (long)arg;
    for (int i = 0; i < loops; i++) {
        sem_wait(&empty);
        put(i + val);
        sem_post(&full);
    }
    return NULL;
}

void *consumer(void *arg)
{

    //for (int i = 0; i < loops * 2; i++) {
    for (int i = 0; i < loops; i++) {
        sem_wait(&full);
        int tmp = get();
        sem_post(&empty);
        printf("%d\n", tmp);
    }
    return NULL;
}

int main()
{
```

```
63      pthread_t p1;
64  //    pthread_t p2;
65      pthread_t c1;
66
67      sem_init(&empty, 0, MAX);
68      sem_init(&full, 0, 0);
69
70      assert(pthread_create(&p1, NULL, producer, (void*)100) == 0);
71  //    assert(pthread_create(&p2, NULL, producer, (void*)200) == 0);
72      assert(pthread_create(&c1, NULL, consumer, NULL) == 0);
73
74      assert(pthread_join(p1, NULL) == 0);
75  //    assert(pthread_join(p2, NULL) == 0);
76      assert(pthread_join(c1, NULL) == 0);
77
78      return 0;
79  }
end
```

```c
// Version 2 - Add mutex before empty/full

// Note: All pthread and sem functions should have their return codes
//       checked. The checking has been omitted to clarity in
//       this example.

#include <assert.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>

#define MAX 10

int buffer[MAX];
int fill = 0;
int use = 0;
int loops = 20;

sem_t empty;
sem_t full;
sem_t mutex;

void put(int value)
{
    buffer[fill] = value;
    fill = (fill + 1) % MAX;
}

int get()
{
    int tmp = buffer[use];
    use = (use + 1) % MAX;
    return tmp;
}

void *producer(void *arg)
{
    int val = (long)arg;

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

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

        put(i + val);

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

    }
    return NULL;
}

void *consumer(void *arg)
{
    for (int i = 0; i < loops; i++) {

        sem_wait(&mutex);
        sem_wait(&full);

        int tmp = get();

```

```
63          sem_post(&empty);
64          sem_post(&mutex);
65
66          printf("%d\n", tmp);
67      }
68      return NULL;
69  }
70
71  int main()
72  {
73      pthread_t p1;
74      pthread_t p2;
75      pthread_t c1;
76
77      sem_init(&mutex, 0, 1);
78      sem_init(&empty, 0, MAX);
79      sem_init(&full, 0, 0);
80
81      assert(pthread_create(&p1, NULL, producer, (void*)100) == 0);
82      assert(pthread_create(&p2, NULL, producer, (void*)200) == 0);
83      assert(pthread_create(&c1, NULL, consumer, NULL) == 0);
84
85      assert(pthread_join(p1, NULL) == 0);
86      assert(pthread_join(p2, NULL) == 0);
87      assert(pthread_join(c1, NULL) == 0);
88
89      return 0;
90  }
end
```

```c
// Version 3 - Add empty/full before mutex

// Note: All pthread and sem functions should have their return codes
//       checked. The checking has been omitted to clarity in
//       this example.

#include <assert.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>

#define MAX 3

int buffer[MAX];
int fill = 0;
int use = 0;
int loops = 20;

sem_t empty;
sem_t full;
sem_t mutex;

void put(int value)
{
    buffer[fill] = value;
    fill = (fill + 1) % MAX;
}

int get()
{
    int tmp = buffer[use];
    use = (use + 1) % MAX;
    return tmp;
}

void *producer(void *arg)
{
    int val = (long)arg;
    for (int i = 0; i < loops; i++) {

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

        put(i + val);

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

    }
    return NULL;
}

void *consumer(void *arg)
{
    for (int i = 0; i < (loops*2); i++) {

        sem_wait(&full);
        sem_wait(&mutex);

        int tmp = get();

        sem_post(&mutex);
```

```
63          sem_post(&empty);
64
65          printf("%d\n", tmp);
66      }
67      return NULL;
68  }
69
70  int main()
71  {
72      pthread_t p1;
73      pthread_t p2;
74      pthread_t c1;
75
76      sem_init(&mutex, 0, 1);
77      sem_init(&empty, 0, MAX);
78      sem_init(&full, 0, 0);
79
80      assert(pthread_create(&p1, NULL, producer, (void*)100) == 0);
81      assert(pthread_create(&p2, NULL, producer, (void*)200) == 0);
82      assert(pthread_create(&c1, NULL, consumer, NULL) == 0);
83
84      assert(pthread_join(p1, NULL) == 0);
85      assert(pthread_join(p2, NULL) == 0);
86      assert(pthread_join(c1, NULL) == 0);
87
88      return 0;
89
90  }
end
```