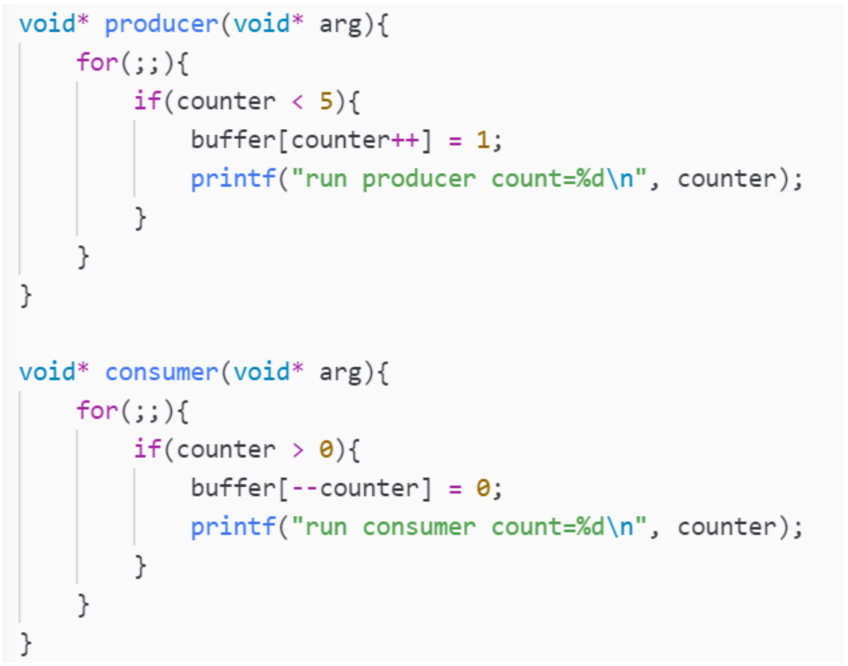
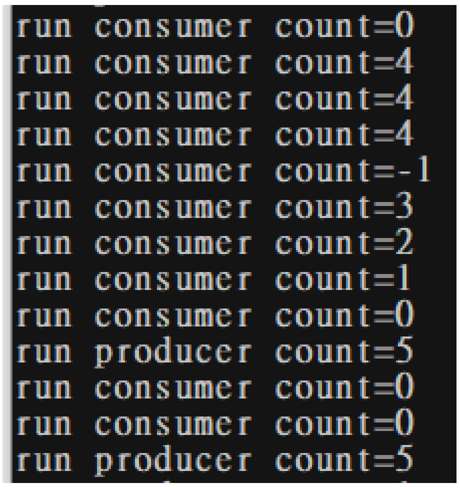
First version



```c
void* producer(void* arg){
    for(;;){
        if(counter < 5){
            buffer[counter++] = 1;
            printf("run producer count=%d\n", counter);
        }
    }
}

void* consumer(void* arg){
    for(;;){
        if(counter > 0){
            buffer[--counter] = 0;
            printf("run consumer count=%d\n", counter);
        }
    }
}
```

Do not use semaphore. Race condition occurs



```
run consumer count=0
run consumer count=4
run consumer count=4
run consumer count=4
run consumer count=-1
run consumer count=3
run consumer count=2
run consumer count=1
run consumer count=0
run producer count=5
run consumer count=0
run consumer count=0
run producer count=5
```

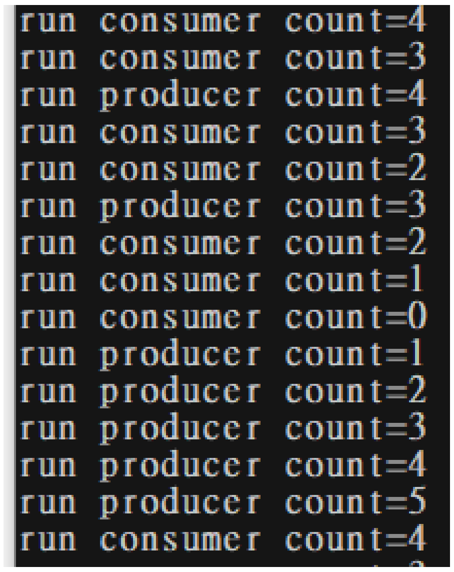
Second Version

use semaphore to prevent race condition

```c
sem_t mutex;

void* producer(void* arg){
    for(;;){
        sem_wait(&mutex);
        if(count < 5){
            buffer[count++] = 1;
            printf("run producer count=%d\n", count);
        }
        sem_post(&mutex);
    }
    pthread_exit(NULL);
}

void* consumer(void* arg){
    for(;;){
        sem_wait(&mutex);
        if(count > 0){
            buffer[--count] = 0;
            printf("run consumer count=%d\n", count);
        }
        sem_post(&mutex);
    }
    pthread_exit(NULL);
}
```



```
run consumer count=4
run consumer count=3
run producer count=4
run consumer count=3
run consumer count=2
run producer count=3
run consumer count=2
run consumer count=1
run consumer count=0
run producer count=1
run producer count=2
run producer count=3
run producer count=4
run producer count=5
run consumer count=4
```