코드 이미지 및 설명(주석으로 작성)

```
int get()
{
    if(get_ptr == put_ptr){ // 버퍼가 비어있으면
        printf("queue is empty!\n");
        return -1; // exit(EXIT_FAILURE)방법도 있다고한다
    }
    get_ptr = (get_ptr + 1) % MAX; // 원형큐 이므로 MAX값으로 나눠서 그 다음 get_ptr위치 조정
    count--; // 버퍼에서 빼서 count값 감소
    return buffer[get_ptr];  // return buffer's value using put_ptr if successful
}
```

```
int put(int val)
{
    if((put_ptr + 1) % MAX == get_ptr){
        printf("buffer is full!\n");
        return -1; // otherwise, -1
    }
    buffer[put_ptr] = val; // 인자로 받은 값 버퍼에 넣기
    put_ptr = (put_ptr + 1) % MAX; // 버퍼안 put_ptr값 MAX로 나눠서 그 다음 put_ptr위치 조정
    count++; // 버퍼에 넣어서 count값 증
    return buffer[put_ptr]; // return buffer's value using put_ptr if successful
}
```

```
void *producer(void *arg)
{
    pthread_mutex_lock(&m_id);
    int id = prod_id++;
    pthread_mutex_unlock(&m_id);
    for (int i = 0; i < PROD_ITEM; ++i) {
        usleep(10);
       /*---------homework---------- */
        pthread_mutex_lock(&m_id); // 다른 스레드 접근 못하게 뮤텍스 락
        while(count == MAX - 1) // 버퍼가 꽉 차면
            pthread_cond_wait(&empty, &m_id); // 소비자로부터 empty큐에 뭐가 있으면, 버퍼에 값을
넣기 위해 lock을 푼다
        int ret = put(i);
        pthread_cond_signal(&fill); // 버퍼에 값을 넣었으므로 fill큐에 신호전달, 소비자가 값 소비
하게 해준다.
        pthread_mutex_unlock(&m_id);
        /* ----------homework---------- */

        if (ret == -1) {
            printf("can't put, becuase buffer is full.\n");
        } else {
            printf("producer %d PUT %d\n", id, ret);
        }
    }
    return NULL;
}
```

```c
void *consumer(void *arg)
{
    pthread_mutex_lock(&m_id);
    int id = cons_id++;
    pthread_mutex_unlock(&m_id);
    for (int i = 0; i < CONS_ITEM; ++i) {
        usleep(10);

        /* ----------homework---------- */
        pthread_mutex_lock(&m_id);
        while (count == 0) { // buffer is emtpy
            pthread_cond_wait(&fill, &m_id); // 프로듀서가 값을 넣기 전까지 기다린다.
        }
        int ret = get();// buffer에 값 생겨서 소비자는 값 얻음
        pthread_cond_signal(&empty); // empty 신호 발생, producer은 값을 buffer로
        pthread_mutex_unlock(&m_id);
        /* ----------homework---------- */


        if (ret == -1) {
            printf("can't get, becuase buffer is empty.\n");
        } else {
            printf("consumer %d GET %d\n", id, ret);
        }
    }
    return NULL;
}
```

결과

```
u201600253@sejung-VirtualBox:~/Desktop/week07$ vi pc_cv_201600253.c
u201600253@sejung-VirtualBox:~/Desktop/week07$ gcc pc_cv_201600253.c -pthread
u201600253@sejung-VirtualBox:~/Desktop/week07$ ./a.out
producer 1 PUT 0
consumer 2 GET 0
producer 2 PUT 0
consumer 1 GET 0
producer 3 PUT 0
consumer 4 GET 0
producer 1 PUT 0
consumer 3 GET 0
producer 2 PUT 0
consumer 5 GET 0
producer 3 PUT 0
consumer 6 GET 0
producer 1 PUT 0
consumer 7 GET 0
producer 2 PUT 0
consumer 1 GET 0
producer 3 PUT 0
consumer 2 GET 0
producer 1 PUT 0
consumer 4 GET 0
producer 2 PUT 0
consumer 3 GET 0
producer 3 PUT 0
consumer 5 GET 0
producer 1 PUT 1
consumer 6 GET 1
producer 2 PUT 1
consumer 7 GET 1
producer 3 PUT 1
```

느낀점

아직도 구현하는게 너무 어렵다. 이론으로 이해하는것은 어렵지 않으나, 코드로 동작

방법을 구현하는 것은 어렵다. 어떻게 해야 더 빠르고 효과적으로 구현하는지 더 연습이 필요할 것 같다.