Implementing producer/consumer with condition variables.

Main Program

```
int max, loops, consumers;
int * buffer;
int fillIndex, useIndex, count = 0;

int main(int argc, char * argv[])

    /* Process arguments */
    max = atoi(argv[1]);
    int loops = atoi(argv[2]);
    consumers = atoi(argv[3]);

    /* Allocate space for the buffer
    buffer = malloc(sizeof(int)*max);

    pthread_t prods, cons[numConsumers];

    /* Create the producer */
    pthread_create(&prods, NULL,
                   producer, NULL);

    /* Create the consumers */
    int i;
    for (i = 0; i < numConsumers; i++)
        pthread_create(&cons[i], NULL,
                       consumer, NULL);


    /* Wait for all threads to finish */
    pthread_join(prods, NULL);
    for (i = 0; i < numConsumers; i++)
        pthread_join(cons[i], NULL);


    return 0;
```

```
/* Buffer Operations */
void put(int value) {
    buffer[fillIndex] = value;
    fillIndex = (fillIndex + 1) % max;
    count++;
}

int get() {
    int tmp = buffer[useIndex];
    useIndex = (useIndex + 1) % max;
    count--;
    return tmp;
}

/* Currently unsafe producer and consumer */
void * producer(void * arg) {
    for (i = 0; i < loops; i++) {
        put(i);
    }
}

void * consumer(void * arg) {
    for (i = 0; i < loops; i++) {
        int tmp = get();
        printf("%d", tmp);
    }
}
```

Solution #1: Single CV, "if"

```
cond_t cond; mutex_t mutex;
void * producer (void * arg) {
    for (i = 0; i < loops; i++) {
        mutex_lock(&mutex);                 // acquire the lock
        if (count == max)                   // check if the buffer is full
            cond_wait(&cond, &mutex);       // wait on the CV
        put(i);                             // produce!
        cond_signal(&cond);                 // signal the CV
        mutex_unlock(&mutex);               // release the lock
}  }
void * consumer (void * arg) {
    for (i = 0; i < loops; i++) {
        mutex_lock(&mutex);                 // acquire the lock
        if (count == 0)                     // check if the buffer is empty
            cond_wait(&cond, &mutex);       // wait on the CV
        int tmp = get();                    // consume!
        cond_signal(&cond);                 // signal the CV
        mutex_unlock(&mutex);               // release the lock
        printf("%d", tmp);
}  }
```

Solution #2: Single CV, "while"

Changes: Replace `if`'s while `while`'s in Solution 1.

Why? State of the buffer might change between when a thread is woken and when it gets to run.

Solution #3: Two CVs

```
cond_t empty, full; mutex_t mutex;
void * producer (void * arg) {
    for (i = 0; i < loops; i++) {
        mutex_lock(&mutex);                 // acquire the lock
        while (count == max)                // check if the buffer is full
            cond_wait(&empty, &mutex);      // wait for buffer to be emptied
        put(i);                             // produce!
        cond_signal(&full);                 // signal that the buffer is full
        mutex_unlock(&mutex);               // release the lock
}  }
void * consumer (void * arg) {
    for (i = 0; i < loops; i++) {
        mutex_lock(&mutex);                 // acquire the lock
        while (count == 0)                  // check if the buffer is empty
            cond_wait(&full, &mutex);       // wait for the buffer to be filled
        int tmp = get();                    // consume!
        cond_signal(&empty);                // signal that the buffer is empty
        mutex_unlock(&mutex);               // release the lock
        printf("%d", tmp);
}  }
```

Changes: Producer threads wait on **empty** and signal **full**. Consumer threads wait on **full** and signal **empty**.

Why? Consumer will never accidentally wake a consumer; producer will never accidentally wake a producer.