MAIN PROGRAM

```
int main(int argc, char *argv[]) {
    max   = atoi(argv[1]);
    loops = atoi(argv[2]);
    consumers = atoi(argv[3]);
    buffer = (int *) Malloc(max * sizeof(int));
    pthread_t pid, cid[CMAX];
    Pthread_create(&pid, NULL, producer, NULL);
    for (int i = 0; i < consumers; i++)
        Pthread_create(&cid[i], NULL, consumer, NULL);
    Pthread_join(pid, NULL);
    for (i = 0; i < consumers; i++)
        Pthread_join(cid[i], NULL);
}
```

QUEUE GET/PUT

```
void do_fill(int value) {
    buffer[fillptr] = value;
    fillptr = (fillptr + 1) % max;
    numfull++;
}

int do_get() {
    int tmp = buffer[useptr];
    useptr = (useptr + 1) % max;
    numfull--;
    return tmp;
}
```

Solution v1 (Single CV)

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m);          // p1
    while (numfull == max)   // p2
      Cond_wait(&cond, &m);  // p3
    do_fill(i);              // p4
    Cond_signal(&cond);      // p5
    Mutex_unlock(&m);        // p6
  }
}
void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m);          // c1
    while (numfull == 0)     // c2
      Cond_wait(&cond, &m);  // c3
    int tmp = do_get();      // c4
    Cond_signal(&cond);      // c5
    Mutex_unlock(&m);        // c6
    printf("%d\n", tmp);
  }
}
```

Solution v2 (2 CVs, "if")

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m);          // p1
    if (numfull == max)      // p2
      Cond_wait(&empty, &m); // p3
    do_fill(i);              // p4
    Cond_signal(&fill);      // p5
    Mutex_unlock(&m);        // p6
  }
}
void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m);          // c1
    if (numfull == 0)        // c2
      Cond_wait(&fill, &m);  // c3
    int tmp = do_get();      // c4
    Cond_signal(&empty);     // c5
    Mutex_unlock(&m);        // c6
    printf("%d\n", tmp);
  }
}
```

Solution v3 (2 CVs, "while")

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m);          // p1
    while (numfull == max)   // p2
      Cond_wait(&empty, &m); // p3
    do_fill(i);              // p4
    Cond_signal(&fill);      // p5
    Mutex_unlock(&m);        // p6
  }
}
void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m);          // c1
    while (numfull == 0)     // c2
      Cond_wait(&fill, &m);  // c3
    int tmp = do_get();      // c4
    Cond_signal(&empty);     // c5
    Mutex_unlock(&m);        // c6
    printf("%d\n", tmp);
  }
}
```

Solution v4 (2 CVs, "while", unlock)

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m);          // p1
    while (numfull == max)   // p2
      Cond_wait(&empty, &m); // p3
    Mutex_unlock(&m);        // p3a
    do_fill(i);              // p4
    Mutex_lock(&m);          // p4a
    Cond_signal(&fill);      // p5
    Mutex_unlock(&m);        // p6
  }
}
void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m);          // c1
    while (numfull == 0)     // c2
      Cond_wait(&fill, &m);  // c3
    Mutex_unlock(&m);        // c3a
    int tmp = do_get();      // c4
    Mutex_lock(&m);          // c4a
    Cond_signal(&empty);     // c5
    Mutex_unlock(&m);        // c6
  }
}
```