

```

#define N 100                                /* number of slots in the buffer */
int count = 0;                               /* number of items in the buffer */

void producer(void)
{
    int item;

    while (TRUE) {                           /* repeat forever */
        item = produce_item();               /* generate next item */
        if (count == N) sleep();             /* if buffer is full, go to sleep */
        insert_item(item);                   /* put item in buffer */
        count = count + 1;                   /* increment count of items in buffer */
        if (count == 1) wakeup(consumer);    /* was buffer empty? */
    }
}

void consumer(void)
{
    int item;

    while (TRUE) {                           /* repeat forever */
        if (count == 0) sleep();             /* if buffer is empty, got to sleep */
        item = remove_item();                /* take item out of buffer */
        count = count - 1;                   /* decrement count of items in buffer */
        if (count == N - 1) wakeup(producer); /* was buffer full? */
        consume_item(item);                  /* print item */
    }
}

```

Fig. 2-23. The producer-consumer problem with a fatal race condition.