

```

h {
    /* Synchronous version */
    y = f(x); /* Call f (potentially blocking) */
    z = g(y); /* Use the result from f to call g */
    q += z; /* Use the value of z to update q */
    return;
}

```

(a)

```

h1 {
    /* Asynchronous version */
    allocate global variables y, z, and q;
    establish cbf1 as the callback function for f1;
    establish cbg1 as the callback function for g1;
    Start f1(x) with a nonblocking call;
    return;
}

```

```

function cbf1(retval) { /* Callback function for f1 */
    y = retval;
    start g1(y) with a nonblocking call;
    return;
}

```

```

function cbg1(retval) { /* Callback function for g1 */
    z = retval;
    q += z;
    return;
}

```

(b)

Figure 23.7 (a) An example of synchronous code, and (b) the structure of asynchronous code for the same computation. Asynchrony adds complexity for the programmer.