```
sum = 0;
for (i = 0; i < 10; i++) {
  if (i % 2)
    continue;
  sum += i;
}
printf("%d\n", sum);
```

W 12. The following "prime-testing" loop appeared in Section 6.4 as an example:

```
for (d = 2; d < n; d++)
  if (n % d == 0)
    break;
```

This loop isn't very efficient. It's not necessary to divide n by all numbers between 2 and n – 1 to determine whether it's prime. In fact, we need only check divisors up to the square root of n. Modify the loop to take advantage of this fact. *Hint:* Don't try to compute the square root of n; instead, compare d * d with n.

Section 6.5

*13. Rewrite the following loop so that its body is empty:

```
for (n = 0; m > 0; n++)
  m /= 2;
```

W*14. Find the error in the following program fragment and fix it.

```
if (n % 2 == 0);
  printf("n is even\n");
```

Programming Projects

1. Write a program that finds the largest in a series of numbers entered by the user. The program must prompt the user to enter numbers one by one. When the user enters 0 or a negative number, the program must display the largest nonnegative number entered:

```
Enter a number: 60
Enter a number: 38.3
Enter a number: 4.89
Enter a number: 100.62
Enter a number: 75.2295
Enter a number: 0

The largest number entered was 100.62
```

Notice that the numbers aren't necessarily integers.

W 2. Write a program that asks the user to enter two integers, then calculates and displays their greatest common divisor (GCD):

```
Enter two integers: 12 28
Greatest common divisor: 4
```

Hint: The classic algorithm for computing the GCD, known as Euclid's algorithm, goes as follows: Let m and n be variables containing the two numbers. If n is 0, then stop: m contains the GCD. Otherwise, compute the remainder when m is divided by n. Copy n into m and copy the remainder into n. Then repeat the process, starting with testing whether n is 0.