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

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;
```

*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.

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.

3. Write a program that asks the user to enter a fraction, then reduces the fraction to lowest terms:

   ```
   Enter a fraction: 6/12
   In lowest terms: 1/2
   ```

   Hint: To reduce a fraction to lowest terms, first compute the GCD of the numerator and denominator. Then divide both the numerator and denominator by the GCD.

4. Add a loop to the `broker.c` program of Section 5.2 so that the user can enter more than one trade and the program will calculate the commission on each. The program should terminate when the user enters `0` as the trade value:

   ```
   Enter value of trade: 30000
   Commission: $166.00

   Enter value of trade: 20000
   Commission: $144.00

   Enter value of trade: 0
   ```

5. Programming Project 1 in Chapter 4 asked you to write a program that displays a two-digit number with its digits reversed. Generalize the program so that the number can have one, two, three, or more digits. *Hint:* Use a `do` loop that repeatedly divides the number by 10, stopping when it reaches 0.

6. Write a program that prompts the user to enter a number *n*, then prints all even squares between 1 and *n*. For example, if the user enters 100, the program should print the following:

   ```
   4
   16
   36
   64
   100
   ```

7. Rearrange the `square3.c` program so that the `for` loop initializes `i`, tests `i`, and increments `i`. Don't rewrite the program; in particular, don't use any multiplications.

8. Write a program that prints a one-month calendar. The user specifies the number of days in the month and the day of the week on which the month begins:

   ```
   Enter number of days in month: 31
   Enter starting day of the week (1=Sun, 7=Sat): 3

          1  2  3  4  5
    6  7  8  9 10 11 12
   13 14 15 16 17 18 19
   20 21 22 23 24 25 26
   27 28 29 30 31
   ```

   Hint: This program isn't as hard as it looks. The most important part is a `for` statement that uses a variable `i` to count from 1 to `n`, where `n` is the number of days in the month, printing each value of `i`. Inside the loop, an `if` statement tests whether `i` is the last day in a week; if so, it prints a new-line character.

9. Programming Project 8 in Chapter 2 asked you to write a program that calculates the remaining balance on a loan after the first, second, and third monthly payments. Modify the program so that it also asks the user to enter the number of payments and then displays the balance remaining after each of these payments.

10. Programming Project 9 in Chapter 5 asked you to write a program that determines which of two dates comes earlier on the calendar. Generalize the program so that the user may enter any number of dates. The user will enter 0/0/0 to indicate that no more dates will be entered:

```
Enter a date (mm/dd/yy): 3/6/08
Enter a date (mm/dd/yy): 5/17/07
Enter a date (mm/dd/yy): 6/3/07
Enter a date (mm/dd/yy): 0/0/0
5/17/07 is the earliest date
```

11. The value of the mathematical constant e can be expressed as an infinite series:

    $e = 1 + 1/1! + 1/2! + 1/3! + \ldots$

    Write a program that approximates e by computing the value of

    $1 + 1/1! + 1/2! + 1/3! + \ldots + 1/n!$

    where n is an integer entered by the user.

12. Modify Programming Project 11 so that the program continues adding terms until the current term becomes less than ε, where ε is a small (floating-point) number entered by the user.