(d)
```
i = 1; j = 2; k = 3;
printf("%d", (i + 5) % (j + 2) / k);
```

W *2. If `i` and `j` are positive integers, does `(-i)/j` always have the same value as `-(i/j)`? Justify your answer.

3. What is the value of each of the following expressions in C89? (Give all possible values if an expression may have more than one value.)

(a) `8 / 5`
(b) `-8 / 5`
(c) `8 / -5`
(d) `-8 / -5`

4. Repeat Exercise 3 for C99.

5. What is the value of each of the following expressions in C89? (Give all possible values if an expression may have more than one value.)

(a) `8 % 5`
(b) `-8 % 5`
(c) `8 % -5`
(d) `-8 % -5`

6. Repeat Exercise 5 for C99.

7. The algorithm for computing the UPC check digit ends with the following steps:

Subtract 1 from the total.
Compute the remainder when the adjusted total is divided by 10.
Subtract the remainder from 9.

It's tempting to try to simplify the algorithm by using these steps instead:

Compute the remainder when the total is divided by 10.
Subtract the remainder from 10.

Why doesn't this technique work?

8. Would the `upc.c` program still work if the expression `9 - ((total - 1) % 10)` were replaced by `(10 - (total % 10)) % 10`?

Section 4.2

W 9. Show the output produced by each of the following program fragments. Assume that `i`, `j`, and `k` are `int` variables.

(a)
```
i = 7; j = 8;
i *= j + 1;
printf("%d %d", i, j);
```
(b)
```
i = j = k = 1;
i += j += k;
printf("%d %d %d", i, j, k);
```
(c)
```
i = 1; j = 2; k = 3;
i -= j -= k;
printf("%d %d %d", i, j, k);
```
(d)
```
i = 2; j = 1; k = 0;
i *= j *= k;
printf("%d %d %d", i, j, k);
```

10. Show the output produced by each of the following program fragments. Assume that `i` and `j` are `int` variables.

(a)
```
i = 6;
j = i += i;
printf("%d %d", i, j);
```

(b)
```
i = 5;
j = (i -= 2) + 1;
printf("%d %d", i, j);
```

(c)
```
i = 7;
j = 6 + (i = 2.5);
printf("%d %d", i, j);
```

(d)
```
i = 2; j = 8;
j = (i = 6) + (j = 3);
printf("%d %d", i, j);
```

Section 4.3

*11. Show the output produced by each of the following program fragments. Assume that `i`, `j`, and `k` are `int` variables.

(a)
```
i = 1;
printf("%d ", i++ - 1);
printf("%d", i);
```

(b)
```
i = 10; j = 5;
printf("%d ", i++ - ++j);
printf("%d %d", i, j);
```

(c)
```
i = 7; j = 8;
printf("%d ", i++ - --j);
printf("%d %d", i, j);
```

(d)
```
i = 3; j = 4; k = 5;
printf("%d ", i++ - j++ + --k);
printf("%d %d %d", i, j, k);
```

12. Show the output produced by each of the following program fragments. Assume that `i` and `j` are `int` variables.

(a)
```
i = 5;
j = ++i * 3 - 2;
printf("%d %d", i, j);
```

(b)
```
i = 5;
j = 3 - 2 * i++;
printf("%d %d", i, j);
```

(c)
```
i = 7;
j = 3 * i-- + 2;
printf("%d %d", i, j);
```

(d)
```
i = 7;
j = 3 + --i * 2;
printf("%d %d", i, j);
```

W 13. Only one of the expressions `++i` and `i++` is exactly the same as `(i += 1);` which is it? Justify your answer.

Section 4.4

14. Supply parentheses to show how a C compiler would interpret each of the following expressions.

(a) `a * b - c * d + e`
(b) `a / b % c / d`
(c) `- a - b + c - + d`
(d) `a * - b / c - d`

Section 4.5

15. Give the values of `i` and `j` after each of the following expression statements has been executed. (Assume that `i` has the value 1 initially and `j` has the value 2.)

(a) `i += j;`
(b) `i--;`
(c) `i * j / i;`
(d) `i % ++j;`

Programming Projects

1. Write a program that asks the user to enter a two-digit number, then prints the number with its digits reversed. A session with the program should have the following appearance:

```
Enter a two-digit number: 28
The reversal is: 82
```

Read the number using `%d`, then break it into two digits. *Hint:* If `n` is an integer, then `n % 10` is the last digit in `n` and `n / 10` is `n` with the last digit removed.

W 2. Extend the program in Programming Project 1 to handle *three*-digit numbers.

3. Rewrite the program in Programming Project 2 so that it prints the reversal of a three-digit number without using arithmetic to split the number into digits. *Hint:* See the `upc.c` program of Section 4.1.

4. Write a program that reads an integer entered by the user and displays it in octal (base 8):

```
Enter a number between 0 and 32767: 1953
In octal, your number is: 03641
```

The output should be displayed using five digits, even if fewer digits are sufficient. *Hint:* To convert the number to octal, first divide it by 8; the remainder is the last digit of the octal number (1, in this case). Then divide the original number by 8 and repeat the process to arrive at the next-to-last digit. (`printf` is capable of displaying numbers in base 8, as we'll see in Chapter 7, so there's actually an easier way to write this program.)

5. Rewrite the `upc.c` program of Section 4.1 so that the user enters 11 digits at one time, instead of entering one digit, then five digits, and then another five digits.

```
Enter the first 11 digits of a UPC: 01380015173
Check digit: 5
```

6. European countries use a 13-digit code, known as a European Article Number (EAN) instead of the 12-digit Universal Product Code (UPC) found in North America. Each EAN ends with a check digit, just as a UPC does. The technique for calculating the check digit is also similar:

Add the second, fourth, sixth, eighth, tenth, and twelfth digits.
Add the first, third, fifth, seventh, ninth, and eleventh digits.
Multiply the first sum by 3 and add it to the second sum.