Exercises

Section 20.1

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

(a)
```
i = 8; j = 9;
printf("%d", i >> 1 + j >> 1);
```

(b)
```
i = 1;
printf("%d", i & ~i);
```

(c)
```
i = 2; j = 1; k = 0;
printf("%d", ~i & j ^ k);
```

(d)
```
i = 7; j = 8; k = 9;
printf("%d", i ^ j & k);
```

W 2. Describe a simple way to "toggle" a bit (change it from 0 to 1 or from 1 to 0). Illustrate the technique by writing a statement that toggles bit 4 of the variable `i`.

*3. Explain what effect the following macro has on its arguments. You may assume that the arguments have the same type.

```
#define M(x,y)  ((x)^=(y),(y)^=(x),(x)^=(y))
```

W 4. In computer graphics, colors are often stored as three numbers, representing red, green, and blue intensities. Suppose that each number requires eight bits, and we'd like to store all three values in a single long integer. Write a macro named `MK_COLOR` with three parameters (the red, green, and blue intensities). `MK_COLOR` should return a `long` in which the last three bytes contain the red, green, and blue intensities, with the red value as the last byte and the green value as the next-to-last byte.

5. Write macros named `GET_RED`, `GET_GREEN`, and `GET_BLUE` that, when given a color as an argument (see Exercise 4), return its 8-bit red, green, and blue intensities.

W 6. (a) Use the bitwise operators to write the following function:

```
unsigned short swap_bytes(unsigned short i);
```

`swap_bytes` should return the number that results from swapping the two bytes in `i`. (Short integers occupy two bytes on most computers.) For example, if `i` has the value `0x1234` (00010010 00110100 in binary), then `swap_bytes` should return `0x3412` (00110100 00010010 in binary). Test your function by writing a program that reads a number in hexadecimal, then writes the number with its bytes swapped:

```
Enter a hexadecimal number (up to four digits): 1234
Number with bytes swapped: 3412
```

Hint: Use the `%hx` conversion to read and write the hex numbers.

(b) Condense the `swap_bytes` function so that its body is a single statement.

7. Write the following functions:

```
unsigned int rotate_left(unsigned int i, int n);
unsigned int rotate_right(unsigned int i, int n);
```

`rotate_left` should return the result of shifting the bits in `i` to the left by `n` places, with the bits that were "shifted off" moved to the right end of `i`. (For example, the call