4. Modify the `reverse.c` program of Section 8.1 to use the expression `(int) (sizeof(a) / sizeof(a[0]))` (or a macro with this value) for the array length.

W 5. Modify the `interest.c` program of Section 8.1 so that it compounds interest *monthly* instead of *annually*. The form of the output shouldn't change: the balance should still be shown at annual intervals.

6. The prototypical Internet newbie is a fellow named B1FF, who has a unique way of writing messages. Here's a typical B1FF communiqué:

```
H3Y DUD3, C 15 R1LLY C00L!!!!!!!!!!
```

Write a "B1FF filter" that reads a message entered by the user and translates it into B1FF-speak:

```
Enter message: Hey dude, C is rilly cool
In B1FF-speak: H3Y DUD3, C 15 R1LLY C00L!!!!!!!!!!
```

Your program should convert the message to upper-case letters, substitute digits for certain letters (A→4, B→8, E→3, I→1, O→0, S→5), and then append 10 or so exclamation marks. *Hint:* Store the original message in an array of characters, then go back through the array, translating and printing characters one by one.

7. Write a program that reads a 5 × 5 array of integers and then prints the row sums and the column sums:

```
Enter row 1: 8 3 9 0 10
Enter row 2: 3 5 17 1 1
Enter row 3: 2 8 6 23 1
Enter row 4: 15 7 3 2 9
Enter row 5: 6 14 2 6 0

Row totals: 30 27 40 36 28
Column totals: 34 37 37 32 21
```

W 8. Modify Programming Project 7 so that it prompts for five quiz grades for each of five students, then computes the total score and average score for each *student*, and the average score, high score, and low score for each *quiz*.

9. Write a program that generates a "random walk" across a 10 × 10 array. The array will contain characters (all `'.'` initially). The program must randomly "walk" from element to element, always going up, down, left, or right by one element. The elements visited by the program will be labeled with the letters A through Z, in the order visited. Here's an example of the desired output:

```
A . . . . . . . . .
B C D . . . . . . .
. F E . . . . . . .
H G . . . . . . . .
I . . . . . . . . .
J . . . . . . . Z .
K . . R S T U V Y .
L M P Q . . . W X .
. N O . . . . . . .
. . . . . . . . . .
```

Hint: Use the `srand` and `rand` functions (see `deal.c`) to generate random numbers. After generating a number, look at its remainder when divided by 4. There are four possible values for the remainder—0, 1, 2, and 3—indicating the direction of the next move. Before performing a move, check that (a) it won't go outside the array, and (b) it doesn't take us to