```
for (;;) {
  printf("Enter command: ");
  scanf("%d", &cmd);
  switch (cmd) {
    case 0:
      balance = 0.0f;
      break;
    case 1:
      printf("Enter amount of credit: ");
      scanf("%f", &credit);
      balance += credit;
      break;
    case 2:
      printf("Enter amount of debit: ");
      scanf("%f", &debit);
      balance -= debit;
      break;
    case 3:
      printf("Current balance: $%.2f\n", balance);
      break;
    case 4:
      return 0;
    default:
      printf("Commands: 0=clear, 1=credit, 2=debit, ");
      printf("3=balance, 4=exit\n\n");
      break;
  }
}
```

Note that the `return` statement is not followed by a `break` statement. A `break` immediately following a `return` can never be executed. and many compilers will issue a warning message.

6.5 The Null Statement

A statement can be *null*—devoid of symbols except for the semicolon at the end. Here's an example:

```
i = 0; ; j = 1;
```

This line contains three statements: an assignment to `i`. a null statement. and an assignment to `j`.

Q&A

The null statement is primarily good for one thing: writing loops whose bodies are empty. As an example, recall the prime-finding loop of Section 6.4:

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