A: A signal-handling function invoked as a result of `raise` or `abort` may call library functions. `tsignal.c` uses `raise` to invoke the signal handler.

Q: How can `setjmp` modify the argument that's passed to it? I thought that C always passed arguments by value. [p. 636]

A: The C standard says that `jmp_buf` must be an array type, so `setjmp` is actually being passed a pointer.

Q: I'm having trouble with `setjmp`. Are there any restrictions on how it can be used?

A: According to the C standard, there are only two legal ways to use `setjmp`:

- As the expression in an expression statement (possibly cast to `void`).
- As part of the controlling expression in an `if`, `switch`, `while`, `do`, or `for` statement. The entire controlling expression must have one of the following forms, where *constexpr* is an integer constant expression and *op* is a relational or equality operator:

```
setjmp(...)
!setjmp(...)
constexpr op setjmp(...)
setjmp(...) op constexpr
```

Using `setjmp` in any other way causes undefined behavior.

Q: After a program has executed a call of `longjmp`, what are the values of the variables in the program?

A: Most variables retain the values they had at the time of the `longjmp`. However, an automatic variable inside the function that contains the `setjmp` has an indeterminate value unless it was declared `volatile` or it hasn't been modified since the `setjmp` was performed.

Q: Is it legal to call `longjmp` inside a signal handler?

A: Yes, provided that the signal handler wasn't invoked because of a signal raised during the execution of a signal handler. (C99 removes this restriction.)

C99

Exercises

Section 24.1

1. (a) Assertions can be used to test for two kinds of problems: (1) problems that should never occur if the program is correct, and (2) problems that are beyond the control of the program. Explain why `assert` is best suited for problems in the first category.

   (b) Give three examples of problems that are beyond the control of the program.

2. Write a call of `assert` that causes a program to terminate if a variable named `top` has the value `NULL`.