towctrans A call of the `towctrans` function requires two parameters: `wc` (a wide character) and `desc` (a value returned by `wctrans`). `towctrans` maps `wc` to another wide character based on the mapping specified by `desc`. For example, the call

```
towctrans(wc, wctrans("tolower"))
```

is equivalent to

```
towlower(wc)
```

`towctrans` is most useful in conjunction with implementation-defined mappings.

Q & A

Q: How long is the locale information string returned by `setlocale`? [p. 644]

A: There's no maximum length, which raises a question: how can we set aside space for the string if we don't know how long it will be? The answer, of course, is dynamic storage allocation. The following program fragment (based on a similar example in Harbison and Steele's *C: A Reference Manual*) shows how to determine the amount of memory needed, allocate the memory dynamically, and then copy the locale information into that memory:

```
char *temp, *old_locale;

temp = setlocale(LC_ALL, NULL);
if (temp == NULL) {
  /* locale information not available */
}
old_locale = malloc(strlen(temp) + 1);
if (old_locale == NULL) {
  /* memory allocation failed */
}
strcpy(old_locale, temp);
```

We can now switch to a different locale and then later restore the old locale:

```
setlocale(LC_ALL, "");          /* switches to native locale */
...
setlocale(LC_ALL, old_locale); /* restores old locale */
```

Q: Why does C provide both multibyte characters and wide characters? Wouldn't either one be enough by itself? [p. 648]

A: The two encodings serve different purposes. Multibyte characters are handy for input/output purposes, since I/O devices are often byte-oriented. Wide characters, on the other hand, are more convenient to work with inside a program, since every wide character occupies the same amount of space. Thus, a program might