`file_name` points to a string containing a file name. The function should return `true` if the file's extension matches the string pointed to by `extension`, ignoring the case of letters. For example, the call `test_extension("memo.txt", "TXT")` would return `true`. Incorporate the "search for the end of a string" idiom into your function. *Hint:* Use the `toupper` function to convert characters to upper-case before comparing them.

`toupper` function ➤*23.5*

18. Write the following function:

```
void remove_filename(char *url);
```

`url` points to a string containing a URL (Uniform Resource Locator) that ends with a file name (such as `"http://www.knking.com/index.html"`). The function should modify the string by removing the file name and the preceding slash. (In this example, the result will be `"http://www.knking.com"`.) Incorporate the "search for the end of a string" idiom into your function. *Hint:* Have the function replace the last slash in the string by a null character.

Programming Projects

Ⓦ 1. Write a program that finds the "smallest" and "largest" in a series of words. After the user enters the words, the program will determine which words would come first and last if the words were listed in dictionary order. The program must stop accepting input when the user enters a four-letter word. Assume that no word is more than 20 letters long. An interactive session with the program might look like this:

```
Enter word: dog
Enter word: zebra
Enter word: rabbit
Enter word: catfish
Enter word: walrus
Enter word: cat
Enter word: fish

Smallest word: cat
Largest word: zebra
```

Hint: Use two strings named `smallest_word` and `largest_word` to keep track of the "smallest" and "largest" words entered so far. Each time the user enters a new word, use `strcmp` to compare it with `smallest_word`; if the new word is "smaller," use `strcpy` to save it in `smallest_word`. Do a similar comparison with `largest_word`. Use `strlen` to determine when the user has entered a four-letter word.

2. Improve the `remind.c` program of Section 13.5 in the following ways:
   (a) Have the program print an error message and ignore a reminder if the corresponding day is negative or larger than 31. *Hint:* Use the `continue` statement.
   (b) Allow the user to enter a day, a 24-hour time, and a reminder. The printed reminder list should be sorted first by day, then by time. (The original program allows the user to enter a time, but it's treated as part of the reminder.)
   (c) Have the program print a one-*year* reminder list. Require the user to enter days in the form *month*/*day*.

3. Modify the `deal.c` program of Section 8.2 so that it prints the full names of the cards it deals: