**Exercise 1-1.** Run the ``hello, world'' program on your system. Experiment with leaving out parts of the program, to see what error messages you get.

**Exercise 1-2.** Experiment to find out what happens when prints's argument string contains \c, where c is some character not listed above.

**Exercise 1-3.** Modify the temperature conversion program to print a heading above the table.

**Exercise 1-4.** Write a program to print the corresponding Celsius to Fahrenheit table.

**Exercise 1-5.** Modify the temperature conversion program to print the table in reverse order, that is, from 300 degrees to 0.

**Exercsise 1-6.** Verify that the expression getchar() != EOF is 0 or 1.

**Exercise 1-7.** Write a program to print the value of EOF.

**Exercise 1-8.** Write a program to count blanks, tabs, and newlines.

**Exercise 1-9.** Write a program to copy its input to its output, replacing each string of one or more blanks by a single blank.

**Exercise 1-10.** Write a program to copy its input to its output, replacing each tab by \t, each backspace by \b, and each backslash by \\. This makes tabs and backspaces visible in an unambiguous way.

**Exercise 1-11.** How would you test the word count program? What kinds of input are most likely to uncover bugs if there are any?

**Exercise 1-12.** Write a program that prints its input one word per line.

**Exercise 1-13.** Write a program to print a histogram of the lengths of words in its input. It is easy to draw the histogram with the bars horizontal; a vertical orientation is more challenging.

**Exercise 1-14.** Write a program to print a histogram of the frequencies of different characters in its input.

**Exercise 1.15.** Rewrite the temperature conversion program of [Section 1.2](file:///C:\Users\Valentin\AppData\Local\Temp\Rar$EXa1580.5684\tcpl\chapter1.html#s1.2) to use a function for conversion.

**Exercise 1-16.** Revise the main routine of the longest-line program so it will correctly print the length of arbitrary long input lines, and as much as possible of the text.

**Exercise 1-17.** Write a program to print all input lines that are longer than 80 characters.

**Exercise 1-18.** Write a program to remove trailing blanks and tabs from each line of input, and to delete entirely blank lines.

**Exercise 1-19.** Write a function reverse(s) that reverses the character string s. Use it to write a program that reverses its input a line at a time.

**Exercise 1-20.** Write a program detab that replaces tabs in the input with the proper number of blanks to space to the next tab stop. Assume a fixed set of tab stops, say every n columns. Should n be a variable or a symbolic parameter?

**Exercise 1-21.** Write a program entab that replaces strings of blanks by the minimum number of tabs and blanks to achieve the same spacing. Use the same tab stops as for detab. When either a tab or a single blank would suffice to reach a tab stop, which should be given preference?

**Exercise 1-22.** Write a program to ``fold'' long input lines into two or more shorter lines after the last non-blank character that occurs before the n-th column of input. Make sure your program does something intelligent with very long lines, and if there are no blanks or tabs before the specified column.

**Exercise 1-23.** Write a program to remove all comments from a C program. Don't forget to handle quoted strings and character constants properly. C comments don't nest.

**Exercise 1-24.** Write a program to check a C program for rudimentary syntax errors like unmatched parentheses, brackets and braces. Don't forget about quotes, both single and double, escape sequences, and comments. (This program is hard if you do it in full generality.)