`scanf` "peeks" at the final new-line character without actually reading it. This new-line will be the first character read by the next call of `scanf`.

What rules does `scanf` follow to recognize an integer or a floating-point number? When asked to read an integer, `scanf` first searches for a digit, a plus sign, or a minus sign; it then reads digits until it reaches a nondigit. When asked to read a floating-point number, `scanf` looks for

a plus or minus sign (optional), followed by

a series of digits (possibly containing a decimal point), followed by

an exponent (optional). An exponent consists of the letter `e` (or `E`), an optional sign, and one or more digits.

The `%e`, `%f`, and `%g` conversions are interchangeable when used with `scanf`; all three follow the same rules for recognizing a floating-point number.

Q&A

When `scanf` encounters a character that can't be part of the current item, the character is "put back" to be read again during the scanning of the next input item or during the next call of `scanf`. Consider the following (admittedly pathological) arrangement of our four numbers:

```
1-20.3-4.0e3¤
```

Let's use the same call of `scanf` as before:

```
scanf("%d%d%f%f", &i, &j, &x, &y);
```

Here's how `scanf` would process the new input:

- Conversion specification: `%d`. The first nonblank input character is `1`; since integers can begin with `1`, `scanf` then reads the next character, `-`. Recognizing that `-` can't appear inside an integer, `scanf` stores `1` into `i` and puts the `-` character back.
- Conversion specification: `%d`. `scanf` then reads the characters `-`, `2`, `0`, and `.` (period). Since an integer can't contain a decimal point, `scanf` stores –20 into `j` and puts the `.` character back.
- Conversion specification: `%f`. `scanf` reads the characters `.`, `3`, and `-`. Since a floating-point number can't contain a minus sign after a digit, `scanf` stores 0.3 into `x` and puts the `-` character back.
- Conversion specification: `%f`. Lastly, `scanf` reads the characters `-`, `4`, `.`, `0`, `e`, `3`, and ¤ (new-line). Since a floating-point number can't contain a new-line character, `scanf` stores -4.0×10^3 into `y` and puts the new-line character back.

In this example, `scanf` was able to match every conversion specification in the format string with an input item. Since the new-line character wasn't read, it will be left for the next call of `scanf`.