337 Interpreting Control Sequences

Virtually all text-mode terminals are special-purpose computer systems, including a serial port (for communication with a modem or another computer system), a keyboard, a CRT, and of course, a microprocessor, some RAM, and a control program in ROM.

When a character arrives at the terminal, either from the keyboard or the serial port, the terminal's software classifies it as either a display character (which is to be displayed on the CRT) or as a character that introduces a control sequence. A control sequence is used to direct the terminal to do such things as clear the screen, move the cursor in a specified manner, or perhaps change fonts.

In this problem assume you are writing the software for a small terminal with a 10-row, 10-column display (perhaps for a point-of-sale terminal). Rows and columns are numbered 0 through 9. The character that introduces a control sequence is ^, the circumflex. The character (or in one case, the two characters) immediately following the control sequence introducer will direct your software in performing its special functions. Here is the complete list of control sequences you will need to interpret:

`^b` Move the cursor to the beginning of the current line; the cursor row does not change

`^c` Clear the entire screen; the cursor row and column do not change

`^d` Move the cursor down one row if possible; the cursor column does not change

`^e` Erase characters to the right of, and including, the cursor column on the cursor's row; the cursor row and column do not change

`^h` Move the cursor to row 0, column 0; the image on the screen is not changed

`^i` Enter insert mode (see below)

`^l` Move the cursor left one column, if possible; the cursor row does not change

`^o` Enter overwrite mode (see below)

`^r` Move the cursor right one column, if possible; the cursor row does not change

`^u` Move the cursor up one row, if possible; the cursor column does not change

`^^` Write a circumflex (^) at the current cursor location, exactly as if it was not a special character; this is subject to the actions of the current mode (insert or overwrite)

`^ ##` Move the cursor to the row and column specified; # represents a decimal digit; the first # represents the new row number, and the second # represents the new column number

No illegal control sequences will ever be sent to the terminal. The cursor cannot move outside the allowed screen locations (that is, between row 0, column 0 and row 9, column 9).

When a normal character (not part of a control sequence) arrives at the terminal, it is displayed on the terminal screen in a manner that depends on the terminal mode. When the terminal is in overwrite mode (as it is when it is first turned on), the received character replaces the character at the cursor's location. But when the terminal is in insert mode, the characters to the right of and including the cursor's location are shifted right one column, and the new character is placed at the cursor's location; the character previously in the rightmost column of the cursor's row is lost. Regardless of the mode, the cursor is moved right one column, if possible.

Input

The input will contain multiple tests of your terminal software. Each test begins with a line containing an integer N. Following this line there will be N more lines of data, each character of which is to be treated as if it was input, in the order read, to your terminal software. There will be no tab characters in the input data, and ends of lines in the input are to be ignored. Note that blanks in the input data are normal characters to be displayed on your terminal's screen. The last test will be followed by a single line containing the integer 0. No control sequence will be split between two lines of the input data.

At the beginning of each test case you are to assume the terminal screen is clear (that is, filled with blanks), that the terminal is in overwrite mode, and that the cursor is in row 0, column 0 of the screen.

Output

For each input test case, output a line with the case number (these are numbered sequentially starting with 1) and the screen image the way it would look at the end of processing the data in the test case. Enclose the screen image in a "box;" see the sample below for illustration of the required format.

Sample Input

```
7
This is bad^h^c
^05^^
^14/ \^d^b   /   \
^u^d^d^l^l^l^l^l^l^l^l^l
^r^r< ACM >^l^l^d/^b   \
^b^d    \ /
^d^l^lv
7
^i9^l8^l7^l6^l5^l4^l3^l2^l1^l0
^o^d^lThis is #1^d^bThis is #2
^d^bThis is #3^d^bThis is #4
^d^bThis is #5^d^bThis is #6
^d^bThis is #7^d^bThis is #8
^i^d^bThis is #9^d^bThis is #10
^54^e Hello^d^l^l^l^lWorld
0
```

Sample Output

```
Case 1
+----------+
|     ^    |
|    / \   |
|   /   \  |
|  < ACM > |
|   \   /  |
|    \ /   |
|     v    |
|          |
|          |
|          |
+----------+
Case 2
+----------+
|0123456789|
|This is #1|
|This is #2|
|This is #3|
|This is #4|
|This Hello|
|This World|
|This is #7|
|This is #8|
|This is #0|
+----------+
```