11956 Brainfuck

Recently your friend Bob has bought a new brainfuck programmable LED display. However executing a program directly on LED display takes huge amount of time. Because of this, now Bob decided to write the interpreter of display instruction set in order to test and debug all his programs on his PC and only after that execute his code on LED display. However Bob knows only one programming language — its certainly brainfuck (otherwise he would not have bought this LED display). So he asks you to write him an interpreter.

A display's program is a sequence of commands executed sequentially. The commands of the display processor is a subset of brainfuck language commands. The commands that processor was capable to execute were '`>`', '`<`', '`+`', '`-`' and '`.`', which are described in the table below. Moreover, this LED display has an array of 100 bytes of circular memory (initialised with zeros) and a pointer to this array (initialised to point to the leftmost byte of the array). This means, that after incrementing a pointer, which points to the rightmost byte of memory, it will point to the leftmost byte and vice versa. Individual bytes are circular as well, so increasing a 255 gives a 0 and vice versa.

Command	**Description**
`>`	Increment the pointer (to point to the next cell to the right).
`<`	Decrement the pointer (to point to the next cell to the left).
`+`	Increment (increase by one) the byte at the pointer.
`-`	Decrement (decrease by one) the byte at the pointer.
`.`	Output the value of the byte at the pointer.

Input

There is a number of tests T ($T \leq 100$) on the first line. After T tests follow. Each test case is a sequence of display processor commands on a separate line. You can assume that line length is less than 100000.

Output

For each test output a single line '`Case` T`:` D'. Where T is the test number (starting from 1) and D is display's memory dump in hexademical numeration system after executing given brainfuck program. Every byte must be separated exactly by one space character. See examples for clarification. Please note, that the sample input and output is divided into several lines only for convenience.

Sample Input

```
1
..++<><<+++>>++++++++++++++++++++++++++>>>+++
<+...++<><<+++>>++++++++++++++++++++++++++>>>
+++<+...++<><<+++>>++++++++++++++++++++++++++
>>>+++<+.
```

Sample Output

```
Case 1: 1F 00 20 03 1D 03 01 03 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 03 00
```