Certain letter keys on a keyboard are broken but all non-letter keys are fully functional. Yet we have to type a text and we would like to know how many complete lines of the text we can type. Additionally, we would like to know which other letter keys can be broken such that we still can type the same lines from the text.

Input

Your program should read input from file named `keyboard.dat`. The input file includes a number of cases. The first line of each case lists a sequence of letters whose keys are broken. The following lines contain the text that we would like to type. This text ends with the following line

```
END
```

which should be also processed. No line of input is longer than 80 characters. The input is terminated by a case whose first line is

```
finish
```

and this case is not to be processed.

Output

The output should be formatted as a table, see the sample output. The header of the table occupies 4 lines and is shown below following an additional line numbering character positions; this line is not a part of the header.

```
123456789012345678901234567890123456789012345678901234567890123456789
+----------+----------------+-----------------------------+
| Keyboard | # of printable | Additionally, the following |
|          |      lines     |  letter keys can be broken  |
+----------+----------------+-----------------------------+
```

In the remaining rows of the table, you should output two lines for each case of input in the following format, see also the sample output.

- The first column of the table reports the number of the keyboard. Keyboards are numbered sequentially, starting with 1. This number should be printed right adjusted at characters 5-7 of the line. There will be no more that 999 keyboards in the input file.
- The second column of the table reports how many lines of the text of this case can be typed with our broken keyboard. This number should be printed right adjusted at characters 19-21 of the line. There will be no more that 999 lines of text in each case.
- The third column of the table reports which additional letter keys can be broken such that we still can type the lines reported in column two. The additional letter keys should be listed as a sequence of small letters in alphabetic order starting at the 31st character of the line.
- The second line of output for each case is a horizontal line as shown in the sample output.

The only white-space characters used in the table are space and new-line.

Sample Input

```
Xyz
We will work with a basic time unit of an eighth-note. At any
given time, your left foot and right foot will each be on distinct
arrows. Only one foot may perform an action (changing arrows and/or
tapping) during any time unit; jumping is not allowed. Also, you must
remain facing forward in order to see the screen. This puts
limitations on which feet you can use to hit which arrows. Finally,
hitting two arrows in a row with the same foot ("double-tapping") is
exhausting, because you can't shift your weight onto that
foot. Ideally, you want to alternate feet all the way through a string
of consecutive arrows.
END
life
is
sometimes
tough
END
but
one
must
not
give
up
END
finish
```

Sample Output

```
+----------+----------------+-----------------------------+
| Keyboard | # of printable | Additionally, the following |
|          |      lines     |  letter keys can be broken  |
+----------+----------------+-----------------------------+
|     1    |        4       | jkq                         |
+----------+----------------+-----------------------------+
|     2    |        1       | abcdjkmnpqrsvwxyz           |
+----------+----------------+-----------------------------+
|     3    |        3       | acfhjklmpqrswxyz            |
+----------+----------------+-----------------------------+
```