Problem A. ASCII Automata Art

Time limit:	3 seconds
Memory limit:	512 megabytes

This problem statement is quite wordy by itself and does not need a legend. You are given a regular expression and your task is to render its corresponding automaton as an ASCII art text drawing following the specification in the problem statement. Please, see examples.

A regular expression in this problem consists of uppercase letters from `A` to `Z`, special characters `+`, `?`, `*`, and parenthesis that are used for grouping. An input to the problem is given by an ⟨*input*⟩ non-terminal of the following BNF grammar:

⟨*input*⟩ ::= ⟨*expr*⟩

⟨*expr*⟩ ::= ⟨*term*⟩ | ⟨*term*⟩ '`|`' ⟨*expr*⟩

⟨*term*⟩ ::= ⟨*atom*⟩ | ⟨*atom*⟩⟨*term*⟩ | ⟨*term*⟩⟨*atom*⟩

⟨*atom*⟩ ::= ⟨*letter*⟩ | '`(`' ⟨*expr*⟩ '`)`' | ⟨*atom*⟩ '`+`' | ⟨*atom*⟩ '`?`' | ⟨*atom*⟩ '`*`'

⟨*letter*⟩ ::= '`A`' | '`B`' | ... | '`Z`'

A regular expression is rendered as an ASCII art picture using the precise rules that are given below. They recursively define a unique representation for each regular expression as a rectangular *box* of characters with the specified number of rows and columns. Empty characters of the representation, including trailing ones, must be filled with spaces.

A ⟨*term*⟩ that consists of a sequence of n uppercase letters is rendered as a box of 3 rows and $4+n$ columns using `+` and `-` characters to render a border on the first and the last rows and columns as shown in the example. The production rule for the ⟨*term*⟩ non-terminal in the grammar is intentionally ambiguous. The longest possible sequence of adjacent ⟨*letter*⟩ non-terminals in the regular expression must be grouped into a ⟨*term*⟩ and rendered as a single box. For example, a ⟨*term*⟩ of '`NERC`' is rendered as the following 3×8 box:

```
+------+
+ NERC +
+------+
```

A ⟨*term*⟩ that consists of a sequence of ⟨*atom*⟩ non-terminals and ⟨*term*⟩ non-terminals with letters (as described above) is rendered by laying out the constituent boxes left-to-right, aligned vertically to the top, with 2 columns separating adjacent boxes. The height of the resulting box is equal to the maximum height of the constituent boxes. Each pair of adjacent boxes is joined by rendering `->` characters on the 2nd row in the columns between them. For example, a ⟨*term*⟩ of '`N(E)RC`' (consisting of a sequence: ⟨*atom*⟩ of '`A`', ⟨*atom*⟩ of '`(E)`', and a letters-only ⟨*term*⟩ of '`RC`') is rendered as the following 3×20 box:

```
+---+  +---+  +----+
+ N +->+ E +->+ RC +
+---+  +---+  +----+
```

An ⟨*expr*⟩ that consists of a single ⟨*term*⟩ is rendered as its ⟨*term*⟩.

An ⟨*expr*⟩ that consists of a '`|`'-separated sequence of ⟨*term*⟩ non-literals is rendered by laying out the corresponding ⟨*term*⟩ boxes top-to-bottom, aligned to the left, with a single row separating adjacent ⟨*term*⟩ boxes. The width of the resulting box is equal to the maximum width of the ⟨*term*⟩ boxes plus 6. There are 3 additional columns on the left, and 3 on the right. The first column and the last column use `+` and `|` characters to join the 2nd rows of all the ⟨*term*⟩ boxes from the top to the bottom one, with `+` placed on the 2nd row of each ⟨*term*⟩ box. The 2nd and the 3rd columns on the left and the 3rd-to-last

and the 2nd-to-last columns on the right have `->` characters on the 2nd rows of the corresponding ⟨*term*⟩ boxes. Additionally, shorter ⟨*term*⟩ boxes are connected on the right with extra `-` characters on their 2nd rows. For example, an ⟨*expr*⟩ of '`C|ON|TEST`' is rendered as the following 11×14 box:

```
   +---+
+->+ C +---->+
|  +---+     |
|            |
|  +----+    |
+->+ ON +--->+
|  +----+    |
|            |
|  +------+  |
+->+ TEST +->+
   +------+
```

An ⟨*atom*⟩ of '`(`' ⟨*expr*⟩ '`)`' is rendered as its ⟨*expr*⟩.

An ⟨*atom*⟩ of ⟨*atom*⟩ '`+`' is rendered as a box of its source ⟨*atom*⟩ with 2 additional rows at the bottom and 6 additional columns (3 on the left and 3 on the right). The first and the last columns, starting with the 2nd row, and the last row are filled with the connecting characters as shown in the example.

- The 2nd row starts with `+->` and ends with `->+` to connect to the 2nd row of the source ⟨*atom*⟩ box.
- The last row starts with `+<-` to represent a backwards edge in the automaton.

For example, an ⟨*atom*⟩ of '`A+`' is rendered as the following 5×11 box.

```
   +---+
+->+ A +->+
|  +---+  |
|         |
+<--------+
```

An ⟨*atom*⟩ of ⟨*atom*⟩ '`?`' is rendered as a box of its source ⟨*atom*⟩ with 3 additional rows at the top and 6 additional columns (3 on the left and 3 on the right). The first and the last columns (from the 2nd to the 5th row) and the 2nd row are filled with the connecting characters as shown in the example.

- The first row of ⟨*atom*⟩ '`?`' is always empty (filled with spaces).
- The 2nd row ends with `->+` to represent an epsilon-edge in the corresponding automaton.
- The 5th row starts with `+->` and ends with `->+` to connect to the 2nd row of the source ⟨*atom*⟩ box.

For example, an ⟨*atom*⟩ of '`B?`' is rendered as the following 6×11 box.

```
+-------->+
|         |
|  +---+  |
+->+ B +->+
   +---+
```

An ⟨*atom*⟩ of ⟨*atom*⟩ '`*`' is rendered as a box of its source ⟨*atom*⟩ with 5 additional rows (3 at the top and 2 at the bottom) and 6 additional columns (3 on the left and 3 on the right). The first and the last columns, with the 2nd and the last row, are filled with the connecting characters as shown in the example.

- The first row of ⟨*atom*⟩ '`*`' is always empty (filled with spaces).
- The 2nd row ends with `->+` to represent an epsilon-edge in the corresponding automaton.
- The 5th row starts with `+->` and ends with `->+` to connect to the 2nd row of the source ⟨*atom*⟩ box.
- The last row starts with `+<-` to represent a backwards edge in the automata.

For example, an ⟨*atom*⟩ of '`C*`' is rendered as the following 8×11 box.

```
+-------->+
|         |
|  +---+  |
+->+ C +->+
|  +---+  |
|         |
+<--------+
```

An $\langle input \rangle$ is rendered as a box that has 6 more columns than the corresponding box of the $\langle expr \rangle$, with 3 additional columns on the left, and 3 on the right. The 2nd row starts with `S->` to represent the starting state of the automaton and ends with `->F` to represent the final state of the automaton. See the example output.

Input

The input consists of a single line that corresponds to the $\langle input \rangle$ non-terminal of the grammar given the problem statement and has at most 100 characters in length.

Output

On the first line of the output, write two integers h and w — the height and the width, correspondingly, of the $h \times w$ box that corresponds to the given $\langle input \rangle$. On each of the next h lines, write w characters of the corresponding ASCII art rendering.

Example

standard input

```
NE?(ER)C++|(IS)*?|(CHA((LL))ENGING)
```

standard output

```
23 57
      +---+               +----+        +---+
S->+->+ N +->+-------->+->+ ER +->+->+->+ C +->+->+->+->F
   |  +---+  |         |  +----+  |  |  +---+  |  |  |
   |         |  +---+  |          |  |         |  |  |
   |         +->+ E +->+          |  +<--------+  |  |
   |            +---+             |               |  |
   |                              +<--------------+  |
   |                                                 |
   |                                                 |
   +->+--------------->+----------------------------->+
   |  |                |                              |
   |  |                |                              |
   |  +->+--------->+->+                              |
   |     |          |                                 |
   |     |  +----+  |                                 |
   |     +->+ IS +->+                                 |
   |     |  +----+  |                                 |
   |     |          |                                 |
   |     +<---------+                                 |
   |                                                 |
   |  +-----+  +----+  +--------+                     |
   +->+ CHA +->+ LL +->+ ENGING +------------------->+
      +-----+  +----+  +--------+
```