2122 Recognizing S Expressions

The functional programming language LISP uses *s-expressions* to represent programs and the data on which they operate. A simplified definition of an s-expression is as follows.

1. Any single alphabetic letter, upper or lower case, is an s-expression.
2. If u and v are s-expressions, then so is (u, v).

For example, each of the letters '`a`', '`c`' and '`m`' is individually a legal s-expression, as confirmed by part 1 of the definition. Using part 2 of the definition, we determine that '`(a,c)`' is also a legal s-expression, and then so is '`((a,c),m)`'.

Your task, in this problem, is to develop a recognizer for s-expressions of the form just described. For each potential s-expression given in the input, you are to determine if it is legal and to display your finding.

Input

The input data will contain multiple candidates for s-expressions. Each line of the input will contain an s-expression candidate. There may be blanks (spaces) before, after, or both before and after the candidate. No input line will contain more than 72 characters. A blank line (that is, a line containing only zero or more blanks) will follow the last candidate.

Output

For each candidate, display the input line number (starting with 1), the candidate (without any leading blanks), and an indication of whether it is an s-expression or not. Display one blank line between the output for each candidate. Your output should resemble the format shown below.

Sample Input

```
a
  b
    (a,b)
((a,b),(C,d))
  (ab,c)
     [ a , b ]
(This line is blank)
```

Sample Output

```
1: a
   is an s-expression.

2: b
   is an s-expression.

3: (a,b)
   is an s-expression.
```

```
4: ((a,b),(C,d))
   is an s-expression.

5: (ab,c)
   is not an s-expression.

6: [ a , b ]
   is not an s-expression.
```