The task is to develop a 'little google' program - a program to search for words in a text.

Assume a text in the standard input. The text is made of words (upper and lower case letters) and word separators (anything else - spaces, tabs, newlines, punctuation, digits, ...). The text is organized into lines, the text is provided in the standard input. The program stores the text into its memory structures. The text input is terminated by an empty line.

The text is followed by a list of queries. Each query is passed on a single line. A query consists of one or more words (upper and lower case letters), the words are separated by separators (spaces, tabs, newlines, punctuation, digits, ...). The program processes the query and finds all positions in the text with the words from the query. The searching compares only the words, all other characters are ignored in the search. The searching is case insensitive. Moreover, the words from the quire are found if:

- all words from the query are found in the text,
- the order of the words matches the order from the query,
- there are not any extra words inserted between the words from the query,
- the words from the query are matches as whole words.

A query may look like:

```
Lorem ipsum
```

Examples of matched text:

```
basic lorem ipsum version
```

```
separators are skipped lorem, ipsum
```

```
separators are skipped 'lorem, ipsum'
```

```
separators are skipped lorem   , 123456 : 456456 ipsum
```

```
separators are skipped 12lorem34ipsum56
```

```
searches are case insensitive LOREM ipsum
```

```
the matched words may span over more lines like here lorem
ipsum dolor sit amet
```

Examples of unmatched text:

```
a difference in any of the query words lorem ipsun
```

```
a missing word lorem dolor
```

```
an inserted word lorem amet ipsum
```

```
some word from the query is not matched as a whole word lorem ipsumdolor
```

```
invalid word order ipsum lorem
```

The input of the program is the text to search, an empty line, and a list of queries (one query per input line). The list queries ends with EOF.

The output of the program is found/not found information for each input query. If the words from the query are found, the program prints out a list of lines where the words were found. If the words from the query span over more lines in the text, then the starting line is included in the output list. If the words from the query are found more than once on the same line, then the line number is listed more than once in the output list. The line numbers in the output list form a non decreasing sequence.

The program must detect invalid input. If the input is invalid, the program must display an error message and terminate. The following is considered an invalid input:

- the input text is not terminated by an empty line,
- the input text is empty (contains no words).

If a query is empty (contains no words), the program displays a special message (see below). However, it does not terminate, it just proceeds with the following queries.

The program is tested in a limited environment. Both time and memory is limited. The limits are set such that a correct implementation of a naive algorithm passes all tests. Moreover, there is some extra memory available, thus the program may use the extra memory for some precalculated data (this may be needed for advanced searching algorithms, see below). On the other hand, the memory limit is proportional to the input problem size. Therefore, a simple static allocation (e.g. 1000000 input characters) will not pass. There are bonus tests included in this homework. The first bonus test inputs a long text and a moderate number of queries, the queries do have many potential matches in the text. The second bonus test is similar to the first test, moreover, the number of queries is very high. An efficient searching algorithm and input data preprocessing is required to pass the bonus tests.

Sample program run:

```
Text:
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Mauris dictum facilisis augue. Etiam neque. Morbi scelerisque id
luctus id velit. Praesent id justo in neque elementum ultrices.
lorem
+
ipsum

Searches:
ipsum
Found: 1, 6
lOrEm
Found: 1, 4
id
Found: 2, 3, 3
um
Not found
amet   consectetuer
```

```
Found: 1
elit Mauris
Found: 1
id    ,    luctus
Found: 2
Etiam Morbi
Not found

Invalid query
lorem ipsum
Found: 1, 4
```

```
Text:
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Mauris dictum facilisis augue. Etiam neque. Morbi scelerisque id
luctus id velit. Praesent id justo in neque elementum ultrices.
Invalid input.
```

Advice:

- The sample runs above list both the output of your program (bold face text) and user input (regular text). The bold/regular formatting is included here, in the problem statement page, to increase readability of the listing. Your program must output the text without any additional markup.