Problem 30: Regular Expression Search[1]

Source filename:	`regexpr.(cpp|java)`
Input filename:	`regexpr.in`
Output filename:	`regexpr.out`

You have been using the `vi` editor on Unix when you suddenly realize that the search command no longer works! When you report the problem to your computation center director, he simply states that there are "other matters in the queue" and can't be bothered with fixing `vi` right now. Thus, you decide to fix the problem yourself.

A regular expression search consists of a search string and a body of text within which to perform the search. The search string is delimited by slashes '`/`' and contains any valid character. however, certain characters have special meaning:

`.` (period)	matches a single occurrence of any character
`*` (asterisk)	matches a sequence of one or more of the previous character in the search string
`\` (backslash)	ake the next character in the search string literally

(Note: This problem defines the asterisk differently from that normally used in regular expressions. Normally, the asterisk is used to match zero or more occurrences of a previous character. In this problem, the asterisk is used to match one or more occurrences of a previous character.)

Any printable character is allowed both in the body of text as well as a search string, and the search *is* case sensitive. You may assume that within a search string, a character will always follow a backslash, and a character will always precede an asterisk. In any search, the end-of-line character is ignored.

Write a program that will read a body of text followed by several strings. The body of text will consist of from 1 to 256 lines, each line containing 0 to 80 characters. The body of text will be terminated by a line containing a single asterisk. Thereafter, the input file will contain 1 or more search strings, one per line. Each search string will begin and end with a slash character. However, it is possible for the search string to contain another slash between the delimiters if it is preceded by a backslash (eg. `/\//`). Any characters on the line outside the delimiting slashes should be ignored.

Your program should first echo the entire body of text to the output file, followed by 2 blank lines. For each search string, your program should echo the search string (excluding any characters outside the delimiting slashes), print each line in the body of text for which a match anywhere within the line is found, and then print 2 blank lines. If not lines in the body of text match the search string, the string "`<no lines>`" should be printed. The list of search strings in the input file is terminated by a line containing a single asterisk.

Example Input File (`regexpr.in`)

```
The Unix* system has become quite popular since its inception
in 1969, running on    machines of varying
processing power from microprocessors to mainframes
and providing a common
execution
environment across them.
*
/ a/
/on/
/on /
/o. /
Ignore /o. r/ spurious characters
/o. m/
/o. *m/
/\*/
*
```

[1] This problem first appeared in the ACM South Central Regional Programming Contest, November 20, 1987.

Example Output File (`regexpr.out`)

```
The Unix* system has become quite popular since its inception
in 1969, running on    machines of varying
processing power from microprocessors to mainframes
and providing a common
execution
environment across them.


/ a/
and providing a common
environment across them.


/on/
The Unix* system has become quite popular since its inception
in 1969, running on    machines of varying
and providing a common
execution
environment across them.


/on /
in 1969, running on    machines of varying


/o. /
in 1969, running on    machines of varying
processing power from microprocessors to mainframes


/o. r/
<no lines>


/o. m/
processing power from microprocessors to mainframes


/o. *m/
in 1969, running on    machines of varying
processing power from microprocessors to mainframes


/\*/
The Unix* system has become quite popular since its inception
```