The Backslash Problem

Problem ID: thebackslashproblem
CPU Time limit: 1 second
Memory limit: 1024 MB
Difficulty: 2.1

Author(s): David Sturgill
Source: Baylor Competitive Learning course

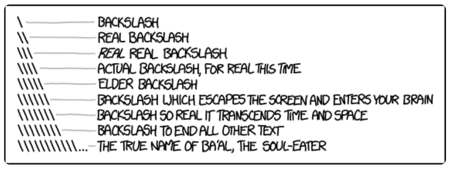
\ BACKSLASH
\\ REAL BACKSLASH
\\\ *REAL* REAL BACKSLASH
\\\\ ACTUAL BACKSLASH, FOR REAL THIS TIME
\\\\\ ELDER BACKSLASH
\\\\\\ BACKSLASH WHICH ESCAPES THE SCREEN AND ENTERS YOUR BRAIN
\\\\\\\ BACKSLASH SO REAL IT TRANSCENDS TIME AND SPACE
\\\\\\\\ BACKSLASH TO END ALL OTHER TEXT
\\\\\\\\\\\\... THE TRUE NAME OF BA'AL, THE SOUL-EATER

Image from xkcd.com/1638

When you are writing programs for a scripting language, you often have to worry about how special characters are handled. If you want to suppress special interpretation and have a special character handled literally, you can escape it by putting a backslash in front of it. In fact, since backslash is, itself, a special character, you also have to escape backslashes if you want them interpreted literally. For example, to get a shell program to print out `*\`, you might have to type something like: `echo \*\\`

Of course, script programmers can't stop at just one level of interpretation. We can write Perl code that outputs shell script, which, itself, might output instructions for something like sed. Each level of interpretation will think that backslashes escape only the character immediately after them. If we want a special character to make it through several levels of interpretation, we may have to place more backslashes in front of it. For example, we could get the shell to print out a single backslash with: ``echo `echo \\\\` ``

Write a program to determine how backslashes should be inserted in order to produce some desired literal string after multiple levels of interpretation. We'll assume that all ASCII characters from `!` up to and including `*` are special. Also, characters from `[` up to and including `]` are considered special. If string s needs to be output after one level of interpretation, we have to start with a string s' where s' is just like s, but with every special character in s escaped. If we want to produce s after two levels of interpretation, we have to start with a string s'' that is just like s', but with every special character in s' escaped.

Input

Input consists of up to 100 test cases. Each test case has two lines. The first line contains an integer $0 \leq n \leq 8$, the number of levels of interpretation that should be applied. The second line contains a string s using only printable ASCII characters, of length 1 to 100. Test cases will end with the end of file.

Output

For each test case, output a string s' containing appropriate backslashes to produce the given string s after n levels of interpretation.

Sample Input 1

```
1
this is a 'test'
2
/:-)
```

Sample Output 1

```
this is a \'test\'
/:-\\\)
```