**Problem B**

**Compressed Words**

Steve has come up with a way to compress text, though it may not actually compress the text. Steve considers only individual words, and uses the following rules to define a "compressed word":

1. a single, lower-case letter is a compressed word.
2. *(e1 e2 ... et n)* where *t* and *n* are non-negative integers and *ei* is a compressed word.

You should observe that a compressed word of one character is the same as an uncompressed word. To uncompress the compressed word *(e1 e2 ... et n)* we uncompress each *ei*, concatenate those uncompressed words into a new word, and repeatedly concatenate that word *n* times. For example:

* x would be uncompressed as x,
* (t 3) would be uncompressed as ttt,
* (a (b c 2) 3) would be uncompressed as abcbcabcbcabcbc.

Write a program to uncompress a compressed word.

**Input**

Your program will be tested on one or more test cases. Each test case is made of one correctly formed compressed word on a separate line. A $ character identifies the end of line. The last line of the input, which is not part of the test cases, contains a $ by itself (possibly with leading and/or trailing white spaces). Every compressed word in the input is correct according to the rules specified above. Note that a compressed word may contain leading, trailing, and/or embedded spaces. Such spaces should be ignored. Letters and numbers are separated from each other by at least one space character.

**Output**

For each test case (i.e., each compressed word), write the uncompressed word on a separate line. There should be no spaces (other than newlines) in the output.

|  |  |
| --- | --- |
| **Sample Input**  x$  (t 3)$  ( a ( b c 2 ) 3) $  $ | **Sample Output**  x  ttt  abcbcabcbcabcbc |