

## 3078 Alphacode

Alice and Bob need to send secret messages to each other and are discussing ways to encode their messages:

Alice: “Let’s just use a very simple code: We’ll assign ‘A’ the code word 1, ‘B’ will be 2, and so on down to ‘Z’ being assigned 26.”

Bob: “That’s a stupid code, Alice. Suppose I send you the word ‘BEAN’ encoded as 25114. You could decode that in many different ways!”

Alice: “Sure you could, but what words would you get? Other than ‘BEAN’, you’d get ‘BEAAD’, ‘YAAD’, ‘YAN’, ‘YKD’ and ‘BEKD’. I think you would be able to figure out the correct decoding. And why would you send me the word ‘BEAN’ anyway?”

Bob: “OK, maybe that’s a bad example, but I bet you that if you got a string of length 500 there would be tons of different decodings and with that many you would find at least two different ones that would make sense.”

Alice: “How many different decodings?”

Bob: “Jillions!”

For some reason, Alice is still unconvinced by Bob’s argument, so she requires a program that will determine how many decodings there can be for a given string using her code.

### Input

Input will consist of multiple input sets. Each set will consist of a single line of digits representing a valid encryption (for example, no line will begin with a 0). There will be no spaces between the digits. An input line of ‘0’ will terminate the input and should not be processed

### Output

For each input set, output the number of possible decodings for the input string. All answers will be within the range of a **long** variable.

### Sample Input

```
25114
1111111111
3333333333
0
```

### Sample Output

```
6
89
1
```