2412 Telephone Alphabetics

Using the telephone pad for alphabetic input.

A telephone set has 12 buttons, as shown in the figure on the right (and some additional buttons, which are not relevant to this problem). On certain occasions (for example, when the user wishes to add a new name to the private phone directory stored in the telephone set) the telephone set is in alphabetical entry mode. In alphabetical entry mode, the user can enter a certain letter by pressing the corresponding digit key as many times as the position (in left-to-right order) of the desired letter on that particular key.

For example, pressing the key 7 once means a 'P', whereas pressing the same key four times means an 'S'. The # key is used to separate successive letters. However, as shown by the examples to be given, pressing the # key between two letters may be omitted when it is clear where one letter ends and the next one begins.

Telephone Keypad

Input

- The input, contains one or more lines, and each line contains one or more characters.
- The only characters (in addition to end-of-line and end-of-file) that may appear in the input are the digits '2' through '9' and the pound character '#'. There will not be leading or trailing blank spaces or tabs in the input.
- The first character of every line will be a digit. The last character (except for end-of-line) of every line will also be a digit.
- The last line of the input will be terminated by end-of-line.
- Each digit or '#' in the input will be interpreted by your program as a keystroke on the above telephone set in alphabetical entry mode.
- The input will not contain two or more consecutive separators (#).

Output

- Every line of input will produce one line of output, consisting of the alphabetical interpretation of the input line (in upper case letters).
- The end-of-line (in addition to the '#' character) will also be interpreted by your program as a separator between letters. In other words, the encoding of a letter will not cross a line boundary in the input.
- The separator '#' between letters may be omitted between two digits that are different. For example, in the first line of the sample input, the separator between 222 and 6 has been omitted, inserting a separator at this point would not affect the output. On the other hand, in the second line of input, you see a redundant separator between 66 and 8. Removing this separator will not affect the output.

- The separator '`#`' between letters may also be omitted in the case of a "long" stream of the same digit: a stream of more than four occurrences of 7, more than four occurrences of 9, or a stream of more than three occurrences of one of the other digits.

  For example, in the third line of the sample input, there is a stream of five 6's. Inserting a separator after the first three 6's would not affect the output. Similarly, in the fourth line of the sample input, removing the separator between the three 6's and the two 6's would not affect the output.

  More generally, in case of a long stream of 6's (or of any other digit other than 7 or 9) a separator could be inserted after every third 6 (starting from the left), without affecting the output.

  For example, '`99996666666666666666`' is equivalent to '`9999666#666#666#666#666#6`', which would produce the output '`ZOOOOOM`'. The same rule applies to long streams of 7's or 9's, with "three" replaced by "four."

- The output will not contain blank lines. Each line of output will contain only upper case letters.

With the sample input below, your program should create the corresponding sample output.

Sample Input

```
2#2226
777666222559996#6668866#8244466
777334#444666662555
7#777666477726#6444664222666#6683377778
```

Sample Output

```
ACM
ROCKYMOUNTAIN
REGIONAL
PROGRAMMINGCONTEST
```