Make Passwords

Ideally a password would be a string of 10 (or more) randomly chosen decimal digits. But this tends to be hard to remember or type.

You are being asked to convert strings of decimal digits into a more typeable and perhaps more memorable form using a 1-1 encoding. In this encoding, groups of more than 3 digits are encoded as 3-letter word randomly chosen from a dictionary.

Some examples are:	161954595198532	*is encoded as*	88bem06fis
	174570539285673	*is encoded as*	5bid012loq
	417744241130258	*is encoded as*	dav034zeq9
	566373677809620	*is encoded as*	sah9qub784
	270281313141987	*is encoded as*	4dud456niz

The encoding algorithm uses a dictionary of all words of the form:

$$< consonant >< vowel >< consonant >$$

sorted in lexical order, with `y` treated as a consonant. Dictionary word 0 is `"bab"` and word $21 * 5 * 21 - 1 = 2204$ is `"zuz"`.

Then the encoding algorithm with input number N is as follows:

1. Divide N by 6 and use the remainder to select the encoding format thus:

0	*selects*	`"WWWDWWWDDD"`	3	*selects*	`"DWWWDDDWWW"`
1	*selects*	`"WWWDDWWWDD"`	4	*selects*	`"DDWWWDDWWW"`
2	*selects*	`"WWWDDDWWWD"`	5	*selects*	`"DDDWWWDWWW"`

2. Process the format left to right. If the next character is `D`, divide N by 10 and output the remainder. If the next character is `W`, divide N by $21*5*21 = 2205$, use the remainder to look up a word in the dictionary, output the word, and skip to after the 3 `W`'s in the format.

For example, $0 + 6 * (0 + 2205 * (1 + 10 * (2204 + 2205 * (2 + 10 * (3 + 10 * 4))))) = 126315290430$ is encoded as `bab1zuz234`. Note you must use '`long`' integers for C, C++, and JAVA, and that numbers input are treated module $6 * (21 * 5 * 21)^2 * 10^4 = 291721500000$.

Input

One more lines each containing a non-negative integer with at most 12 digits. Input ends with an end of line.

Output

For each input line, output one line containing the encoded input integer.

Sample Input

00-000-makepass.sin:

```
0
6
13230
132300
291721500
2917215000
29172150000
```

00-001-makepass.sin:

```
0
1
2
3
4
5
```

00-002-makepass.sin:

```
126315290430
854595198532
870539285673
144241130258
273677809620
281313141987
```

Sample Output

00-000-makepass.sout:

```
bab0bab000
bac0bab000
bab1bab000
bab0bac000
bab0bab100
bab0bab010
bab0bab001
```

00-001-makepass.sout:

```
bab0bab000
bab00bab00
bab000bab0
0bab000bab
00bab00bab
000bab0bab
```

00-002-makepass.sout:

```
bab1zuz234
88pat25yiq
5yuz930zok
vac975yuf4
lod3fan839
4qol723zeh
```

Author: Robert L. Walton <walton@acm.org>
Date: Thu Oct 8 06:35:04 EDT 2020