IDI Open
Programming Contest
April 17th, 2010

Problem Set

A	Guarding the Border
B	Beehive Epidemic
C	Mobile Gaming
D	Balancing Weights (Easy)
E	Ambulance Antics
F	Nurikabe
G	Cookie Monster (Easy)
H	Typing Monkey
I	The Diligent Cryptographer
J	Combat Odds

Jury and Problem Writers

Eirik Reksten, Steria
Ruben Spaans, IDI/NTNU
Tor Gunnar Høst Houeland, IDI/NTNU
Rune Fevang, Opera Software

Problem I

The Diligent Cryptographer

Halvor is in charge of the Single Sign-On (SSO) login system for Identity Directories, Inc. He has been a passionate supporter of their technology for years, telling anyone who will listen how it makes user authentication simpler and more secure with the encrypted login back-end provided by Trustworthy Enterprises (TE). Last week Halvor got a newsletter from TE, where they introduced their new and highly innovative Open Trust Protection (OTP) system, which was recently implemented and has been used for new accounts and users that changed their password in the last month.

Previously, a user's cryptographic key consisted of a permutation of the first letters of the alphabet, repeated many times so it could be used for long messages. In the new improved system the cryptographic key instead consists of random letters generated by a lava lamp-based sub-contractor.

As an example the string `BCAEDBCAEDBCAED` was a possible key in the old system since this is a repetition of `BCAED`, which is a permutation of the letters from `A` to `E`. The strings `BCDBCD` and `BABBABBABBAB` would not be possible, since the letter `A` is missing from the repeated permutation `BCD`, and `BAB` is not a permutation of `AB` since there are two `B`'s.

Halvor decides to change the keys for the users that have not already been automatically moved to the new system. Luckily he has read and write access to all the keys for his users, and has contracted you to write a program to determine which users need to be updated. To avoid any privacy concerns, you are only given a list of the user's names, last login times and up to the first 1000 letters of their key.

Thus for the old system the end of the key substring you receive might be cut off in the middle of a repetition, but the first letter is guaranteed to be the start of a permutation. For the new system the entire string will be random, including the letters you receive.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case consists of one line containing a string K, which is the first part of a user's cryptographic key.

Output specifications

For each test case, output a line containing the line "old" if K is definitely from the old system, "new" if K is definitely from the new system, or "unknown" if this cannot be determined from the provided key substring.

Notes and Constraints

- $1 \leq T \leq 1000$
- $1 \leq |K| \leq 1000$
- *All letters in the input string are uppercase (clarification)*
- The entropy can be written as $H(X) = -\sum_{i=1}^{n} p(x_i) \log_b(x_i)$, where p denotes the probability mass function of X.

Sample input

```
4
ABCD
BB
HELP
IAMTRAPPEDINACRYPTOGRAPHICKEYFACTORY
```

Output for sample input

```
unknown
new
unknown
new
```

IDI Open
Programming Contest
April 2nd, 2011

Problem Set

Jury and Problem Writers

Eirik Reksten, Steria
Ruben Spaans, NTNU
Erik Axel Nielsen, McKinsey & Company
Tor Gunnar Høst Houeland, IDI/NTNU

Problem A

Soundex

Soundex is a phonetic algorithm for transforming a string into a code, which is always a letter followed by three digits. The purpose is that strings with a different spelling but similar pronounciation will be transformed into the same code. The transformation rules are as follows:

- The first letter of the string is the first letter of the code.
- Subsequent consonants are replaced by digits:
  - `b`, `f`, `p`, `v` with 1
  - `c`, `g`, `j`, `k`, `q`, `s`, `x`, `z` with 2
  - `d`, `t` with 3
  - `l` with 4
  - `m`, `n` with 5
  - `r` with 6

  `h` and `w` are ignored. The vowels, `a`, `e`, `i`, `o`, `u` and `y`, are not encoded.
- Two or more adjacent letters with the same digit are replaced with a single digit. Two or more letters with the same digit separated with `h` or `w` are also replaced by a single digit. Two letters with the same digit separated by a vowel will appear as the digit twice.
- Repeat the previous step until it isn't possible to replace repeating digits with one digit.
- If the resulting code has less than 3 digits, pad the end with zeroes until the code has 3 digits. If there are more than 3 digits, drop the rightmost digits.

Some example transformations are:

- Both `robert` and `rupert` are transformed to R163.
- `baawwwww` is transformed to B000.
- `hopp` is transformed to H100. The first letter of the string always becomes the first letter in the code, even if it is a vowel or `h` or `w`.
- `ratatata` is transformed to R333, because the vowel between each pair of `t` (3) forces the digit to be repeated.
- `yhhhwthwhtwhthwhwth` is transformed to X300. All occurrences of `h` and `w` are ignored, leaving only one `3` in the code.
- `bbpb` is transformed to B100. The first `b` is kept separate from the last three `b`'s. The remaining `b`'s are consecutive and are replaced with one digit.

Your evil friend Halvor has noticed that a lot of different words will give the same soundex code. For example, the strings `rhhhbm`, `rubeno`, `rpowam`, `robnew` and 73908 other strings with 6 or less letters will be converted to the code R150. This makes him very curious, and he wants to know the number of strings of the a given length or shorter that will be converted to the same soundex code. Naturally, he wants you to write a computer program to accomplish this task. Your friend does not care about upper or lower case, so `AA`, `Aa`, `aa` etc are considered to be equal strings and should only be counted once.

Input specifications

The first line of input contains a single integer T, the number of test cases to follow. Each test case begins with a line containing a string S and an integer number L. S represents a soundex code, and consists of one uppercase letter followed by three digits. L represents the maximum length of the original string which is converted into the given soundex code.

Output specifications

For each test case output one line containing a single number, the number of strings of length L or less having the soundex code S. This number can be large, so output it modulo 1,000,000,007.

Notes and Constraints

- $0 < T \leq 100$
- $0 < L \leq 1000$
- All soundex codes will start with an uppercase letter.
- Two strings S and T are equal if they have the same length and the letters s_i, t_i at each position are equal, regardless of case.
- Only letters between `a` and `z` are considered. No `æ`, `ø`, `å` or other non-English characters are to be used.
- The soundex code is always legal. That is, a non-zero digit will never follow a zero digit, and the digits range from `0` to `6`.

Sample input

```
3
A300 2
R150 6
X123 1
```

Output for sample input

```
2
73912
0
```