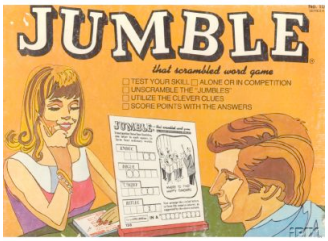
Baldwin-Wallace College

Spring 2008 Programming Contest

Do Not Open Until Instructed



leJmub

A *Jumble*, for our purposes, is defined as an ordering of the letters in a string by one of two techniques: A *Forward-leumJb* or a *Reverse-leJmub*.

The Forward-leumJb of a string is defined as follows:

1. Find the two characters in the string that appear first in the English alphabet. In the event of a tie, use leftmost letters in the current version of string first.
2. Reverse the substring beginning and ending at the points found in step 1 (inclusive, from the one appearing first in the string to the one appearing last in the string). Once a character has been used as an endpoint, it can no longer be used in the following iterations.
3. Repeat this process until all (or all except 1 if the length of the string is odd) characters have been used as endpoints.

The Reverse-leJmub of a string is defined as follows:

1. Find the two characters in the string that appear last in the English alphabet. In the event of a tie, use rightmost letters in the current version of string first.
2. Reverse the substring beginning and ending at the points found in step 1 (inclusive, from the one appearing first in the string to the one appearing last in the string). Once a character has been used as an endpoint, it can no longer be used in the following iterations.
3. Repeat this process until all (or all except 1 if the length of the string is odd) characters have been used as endpoints.

The Forward-leumJb of the string "Jumble" would be (underlined letters are endpoints):
"Jumble" → "Jumelb" → "lemuJb" → "leumJb."

The Reverse-leJmub of the string "Jumble" would be (underlined letters are endpoints):
"Jumble" → "Jmuble" → "lbumJe" → "leJmub."

In a string of length n, there are n*(n-1)/2 pairs of ordered letters in the order they appear within the string. For example, consider the Reverse-leJmub of "cat", which is "tac". The string "tac" has 3 * (2)/2 = 3 ordered pairs of letters: (t, a), (t, c) and (a, c). The *Alphabetic Accuracy* of a Jumble is defined as the number of such ordered pairs in which the characters appear in alphabetic order (if the letters are the same then that pair does NOT count towards the Alphabetic Accuracy). In this example, (a, c) is correctly ordered, whereas (t, a) and (t, c) are incorrectly ordered. Hence, the Alphabetic Accuracy of "tac" (the Reverse-leJmub of "cat"), is 1.

For reference, the English alphabet is as follows: "abcdefghijklmnopqrstuvwxyz"

Details of the Input

The first line of the input will contain a number, *n*, of cases to follow. Each of the following *n* lines will contain a string of characters (of length *len*, $2 \leq len \leq 40$) all falling in the alphabetic range in both upper and lowercase. The case of a letter should have nothing to do with where it falls in alphabetical order for the purposes of this problem.

Details of the Output

For each case, output

```
Case <i>: <Original String> -- <Result>
```

where

<i> represents the case number (in the range [1, *n*])
<Original String> is the string exactly as it appeared in the input
<Result> is one of:
"Reverse-leJmub" – if the reverse-leJmub of the string has greater alphabetic accuracy than the forward-leumJb.
"Forward-leumJb" – if the forward-leumJb of the string has greater alphabetic accuracy than the reverse-leJmub.
"Neither" – if the accuracies are the same for the forward-leumJb and the reverse-leJmub.

There should be exactly one space between "Case" and <i>, between ":" and <Original String>, between <Original String> and "--", and between "--" and <Result>. For clarification, output should match exactly what is shown in the Sample Output below.

Sample Input

```
3
Jumble
cat
GO
```

Sample Output

```
Case 1: Jumble -- Reverse-leJmub
Case 2: cat -- Forward-leumJb
Case 3: GO -- Neither
```

Hoppin' Down the Bunny Trail

Perhaps one of the most confusing holidays to keep track of is Easter[1]. Unlike many holidays where statements like "It occurs on the 25th of December" or "It is always the third Monday of January" apply, Easter is defined to occur on the first Sunday after the first ecclesiastical full moon after the vernal equinox[2]. So when is that anyway?? Well, we all realize it moves around quite a bit, occurring at the earliest on March 22nd (which happened last in 1818 and will happen again in 2285[3]) and latest on April 25th (which happened last in 1943 and will happen again in 2038[2]). What you may not realize is that there are calculations that can be done to compute the date of Easter for a given year. The calculations, credited to mathematician Friedrich Gauss are as follows[4]:

$$a = Y \bmod 19$$
$$b = Y \bmod 4$$
$$c = Y \bmod 7$$
$$d = (19a + M) \bmod 30$$
$$e = (2b + 4c + 6d + N) \bmod 7$$

where values for M and N are determined by the year according to the following table:

Year	M	N
1583-1699	22	2
1700-1799	23	3
1800-1899	23	4
1900-2099	24	5
2100-2199	24	6
2200-2299	25	0

If $d + e < 10$ then Easter is on the $(d + e + 22)$(th, st, nd, rd) of March, and is otherwise on the $(d + e - 9)$(th, st, nd, rd) of April.
The following exceptions must be taken into account:

- If the date given by the formula April 26th, Easter is on April 19th.
- If the date given by the formula is April 25th, with $d = 28$, $e = 6$, and $a > 10$, Easter is on April 18th.

In this problem, you are to implement a program that, given a year in the range 1583-2299, will compute and report the date of Easter. The end of the input will be signified by an input of 0 for the year. Your output must include the proper consonant blend at the end of the date (e.g., 10th vs. 1st) as illustrated in the sample output on the next page.

[1] The dates referred to are under the Gregorian calendar used by most Western churches
[2] http://aa.usno.navy.mil/faq/docs/easter.php
[3] http://en.wikipedia.org/wiki/Easter#Computations
[4] http://en.wikipedia.org/wiki/Computus

Sample Input

```
2008
2050
2038
1976
0
```

Sample Output

```
In 2008 Easter is on March 23rd.
In 2050 Easter is on April 10th.
In 2038 Easter is on April 25th.
In 1976 Easter is on April 18th.
```

On the Fence

One way to secure data is to encrypt it with a cipher. A cipher can be any method or algorithm by which data is essentially jumbled so as to be unreadable by anyone other than the people who know what cipher was used, and how to reverse it.

Charles, an eager yet mischievous student, suspects that his roommate may be trying to extract revenge for a "small" prank involving spaghetti, baby powder, a rabbit, and two pounds of cheese. As such, he has taken to the idea of running all of his personal information through a cipher to make sure it couldn't potentially be used against him. After researching several potential ciphers, and even attempting to make a few on his own, he settled on what is known as a Rail Fence Cipher[5].

Charles has asked for your help in writing a program that would do this. He explains that the Rail Fence Cipher is a form of transposition cipher. That is, it takes the individual characters contained within a particular piece of information, written out in plaintext, and moves them around so that they cannot be read normally. When plaintext is passed through the cipher, it is written downwards and diagonally to the right on successive "rails," When the bottom rail is reached the process reverses the vertical direction now printing upward and diagonally until the top rail is reached. This continues repeatedly until the entire plaintext message has been encrypted. Alphabetic characters will be uppercase only. Non-alphabetic characters do not appear in the encrypted message. Afterwards, the information is read off in rows. Charles provides the following example:

Original Message:

```
WE ARE DISCOVERED. FLEE AT ONCE.
```

As the cipher runs:

```
W . . . E . . . C . . . R . . . L . . . T . . . E
. E . R . D . S . O . E . E . F . E . A . O . C .
. . A . . . I . . . V . . . D . . . E . . . N . .
```

The Final Output:

```
WECRL TEERD SOEEF EAOCA IVDEN
```

[5] http://en.wikipedia.org/wiki/Rail_Fence_Cipher

Details of the Input

There will be multiple input sets. Each set will start with a single positive integer N ($3 \leq N \leq 15$) indicating the number of rails to be used. The next line will contain a string of no more than 70 characters (including blanks) which is to be encrypted with the rail cipher. An N value of 0 indicates the end of the input sets.

Details of the Output

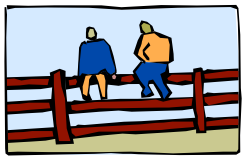
For each line of input there should be one line of output. The characters should be output in groups of 5, with a single space in between. The last group can have fewer than 5 characters, and should be followed by a new line character. There should be no spaces between lines, and no blank lines at the end.

Sample Input

```
3
WE ARE DISCOVERED. FLEE AT ONCE
5
SHELLY SELLS SEA SHELLS BY THE SEA SHORE
4
RELATIVITY PROHIBITS OBJECTS FROM FASTER THAN LIGHT TRAVEL
0
```

Sample Output

```
WECRL TEERD SOEEF EAOCA IVDEN
SLEEE HELHL HSRES SSLTE OLYSA SYAHL EBS
RVOST FTGVE IIRHT OCSMA RHIHA ELTTP IIBEF OSEAL TRLAY BJRTN T
```



Back on the Fence

Charles'[6] roommate, Snidely, realizes that Charles is on to him and has begun intercepting the encrypted messages that Charles is sending. Knowing that you were the mastermind behind the encryption program that Charles is using, Snidely offers you great reward (a pizza a week for the next year) to write a program for him that will decrypt the intercepted messages and you decide it's an offer too good to pass up.

Details of the Input

The first line of the input file is an integer indicating how many messages are to be decrypted. For each message, two lines of input are given. The first line indicates the number of rails that were used in the encryption and the second contains the encrypted message. There will be no blank spaces in the encrypted message.

Details of the Output

Output for each message labels the message on the first line, followed by outputting the decrypted message on rails, using the period character between letters on a rail.

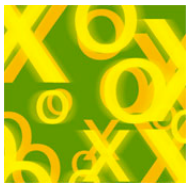
Sample input

```
2
3
WECRLTEERDSOEEFEAOCAIVDEN
3
DWEYUEHRIMCRDESA
```

Sample output

```
Message 1:
W . . . E . . . C . . . R . . . L . . . T . . . E
. E . R . D . S . O . E . E . F . E . A . O . C .
. . A . . . I . . . V . . . D . . . E . . . N . .
Message 2:
D...W...E...Y...
.U.E.H.R.I.M.C.R
..D...E...S...A.
```

[6] This is the same Charles as in On the Fence



Tic-Tac-Trouble

Mark and Julie like to spend rainy afternoons playing friendly games of tic-tac-toe while waiting for the sun to come out. The brother and sister are both fairly good players, but Julie, who is older, usually beats Mark in the game. To check that is his sister is not cheating and really does win, Mark asks their mother to watch them play their games.

One rainy day when Mark and Julie were playing, their Mom left to go shopping. The two were left were their grandmother, who upon arriving at the house, immediately fell asleep in a chair. Julie suggested that they play a slightly modified version of the game. Instead of playing on the usual three-by-three (3x3) grid, they would instead play on a grid of four-by-four (4x4), with the winner determined by the first to get four (4) in a row, column, or along a diagonal. Mark, being skeptical of his sister and her schemes, suspected that a larger grid might give Julie more opportunity to make suspicious moves like replacing one of his moves with hers. To make sure that she wasn't cheating, Marc decided to write down the moves he and his sister made for each game, which he would later show his mother for her to check. When he did show the game information to his mother, she simply pushed it to the side, saying how difficult it was to interpret the information without having watched the game. Mark still wishes to know if the games he played with his sister are fair or not, so he is soliciting your help.

Your job is to write a program that would check the outcome of the games of Tic-Tac-Toe that Mark and Julie played and tell who won, if they tied, or if there was an illegal move made. The game grid will be viewed as indicated below. That is, rows will be indicated by the letters A..D and columns by the numbers 1..4.

Game Grid

A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4
D1	D2	D3	D4

Details of the Input

The first line of input will be an integer, *n*, indicating the number of games that is to be played. For each game, the first input will be an integer indicating the number of moves, *m*, in the game, $1 \leq m \leq 16$. The next line will contain *m* grid names, each separated by a single space. Each grid name represents the move of a player, with turns alternating between players. In all games, Julie is assumed to be the player who moves first ("girls first", she says).

Details of the Output

Each line of the output file reports the outcome of a single game. The output should be one of the following:

- Game *[#]* was won by *[name]*.
- There was no winner in game *[#]*.
- *[name]* made an illegal move in game *[#]*.

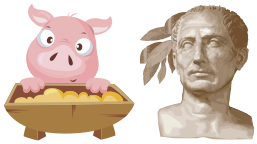
where *[#]* is the number of the current game (1..*n*) and *[name]* is either Mark or Julie.

Sample input

```
2
7
A1 B2 B1 D3 C1 A2 D1
10
A4 A2 B3 B2 C2 A4 D1 A3 D2 A1
```

Sample output

```
Game 1 was won by Julie.
Mark made an illegal move in game 2.
```



Igpay Atinlay

Your seven year old cousin, Bengi, has taken a real interest in Latin, so much so that his every word is an attempt at Latin. Well, this is not exactly the Latin of Julius Caeser, but the second grade version, Pig Latin. Although he has gotten quite good at it, he asked if you could write a program that would translate an entire paragraph into Pig Latin. There are different versions of Pig Latin, but the one used by Bengi's friends uses the following rules:

- Words that start with a vowel ('a', 'e', 'i', 'o', 'u') simply have "way" appended to the end of the word.
- Words beginning with consonants, including 'Y' or 'y', have all consonants up to the first vowel moved to the end of the word and then the phrase "ay" is appended after that
    - In the event the starting consonant is a 'q', the 'u' stays with the 'q' – quit=>itquay
    - The first vowel may be a 'y' – the only case in which 'y' is treated as a vowel
        - yes => esyay
        - style => ylestay
- Subparts of hyphenated words are converted independently – mish-mash -> ishmay-ashmay
- If a word was capitalized, the first letter of the Pig Latin word should be capitalized.
- If a word had punctuation at the beginning or end (e.g., 'Twas, house., or "quote"), the punctuation should appear at the end of the Pig Latin word ('Astway, ousehay., or "otequay"). Only single-character punctuation marks will be used and now quoted word will appear before a punctuation mark.

Your program is to translate a paragraph of text into Pig Latin. The input paragraph will contain an unknown number of words, but the last word in the input will be 'zzzzz'. 'zzzzz' should not appear in the output. Each line of output (except possibly the last) should contain exactly six words separated by a single blank. Hyphenated words count as only one word. The input will not contain any contractions, or possessives. For example, the possessive of class (class') or Johnny (Johnny's) will not appear.

Sample Input

```
Here is some sample input for you to try out. Bengi hopes you
can do this! zzzzz
```

Sample Output

```
Erehay isway omesay amplesay inputway orfay
ouyay otay rytay outway. Engibay opeshay
ouyay ancay oday histay!
```