الف) زبان اعداد

در زمان جنگ جهانی اول که با توجه به شواهد موجود یکی از علل وقوع آن را به آقای خالقی نسبت می دهند، یکی از روش‌های که نظامیان با هم صحبت می‌کردند زبان اعداد بود، که هر عدد نشان دهنده حرف یا کلمه خاصی بودند.

a 1
b 2
...
z 26
aa 27
ab 28
...
snowfall 157,118,051,752
...

در این جنگ مجتبی یک مترجم است که حرف یا کلمه را به عدد تبدیل می‌کند و به‌عکس. برنامه‌ای بنویسید که کار مترجم را انجام دهد.

اطلاعات ورودی:

-متن وارد شده برای ترجمه. و در صورتی که * باشد نشان دهنده پایان ورود اطلاعات است.

اطلاعات خروجی:

-تبدیل جملات

Sample Input:

```
29697684282993
transcendental
2801162263682385445620
computationally
zzzzzzzzzzzzzzzzzzzz
*
```

Sample Output:

```
elementary              29,697,684,282,993
transcendental          51,346,529,199,396,181,750
prestidigitation        28,011,622,636,823,854,456,520
computationally         232,049,592,627,851,629,097
zzzzzzzzzzzzzzzzzzzz    20,725,274,851,017,785,518,433,805,270
```

ب) جمع

جمع یکی از ساده‌ترین عمل‌های ریاضیات بوده. حمید که دوست دارد دوستان خود را سرگرم کند. بازی طراحی کرده است که دو عدد(تعداد سطر و ستون جدول) را به دوستان خود می‌دهد و از آنها می‌خواهد جمع اعداد داخل جدول را محاسبه کنند. این جدول به صورت قطری از صفر پُر می‌شوند. مانند:

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

برنامه‌ای بنویسید که این کار را انجام دهد.

اطلاعات ورودی:

-تعداد آزمون ها $1 \le t \le 100$

- اندازه سطر و ستون جدول($1 \le n \le 100$،$1 \le m \le 100$)

اطلاعات خروجی:

-مجموع اعداد داخل جدول

Sample Input:

```
3
2 2
2 3
3 4
```

Sample Output:

```
4
9
30
```

پ) شمارش اعداد

صالح می‌خواهد یک لیست از اعداد بین دو عدد دلخواه مثبت را بنویسد. او قصد دارد تا تعداد تکرار هر رقم را پیدا کند.

اطلاعات ورودی:

-دو عدد دلخواه A وB (در صورتی که دو عدد دلخواه صفر باشند پایان اجرای برنامه)$(1 \le A \le B \le 10^8)$.

اطلاعات خروجی:

-برای هر دو عدد دلخواه تعداد رقم های استفاده شده در لیست.

Sample Input:

```
1 9
12 321
5987 6123
12345678 12345679
0 0
```

Sample Output:

```
0 1 1 1 1 1 1 1 1
61 169 163 83 61 61 61 61 61
134 58 28 24 23 36 147 24 27 47
0 2 2 2 2 2 2 1 1
```

ت) تقسیم

علی که معلوم نیست در کجا به سر می‌برد، بازی طراحی کرده است مجموع مقسوم علیه های یک عدد را با هم جمع می‌کند. به عنوان مثال مقسوم‌علیه های عدد ۱۰ (۱،۲،۵) هستند، که مجموع آنها برابر ۸ می شود.

برنامه ای بنویسید که عدد مثبت را گرفته سپس مجموع را حساب کند.

اطلاعات ورودی:

-تعداد مراحل آزمون

-عدد صحیح و مثبت برای محاسبه مجموع مقسوم علیه.(1 <= n <= 500000)

اطلاعات خروجی:

-مجموع مقسوم‌علیه برای هر ورودی

Sample Input:

```
3
2
10
20
```

Sample Output:

```
1
8
22
```

E) URLs

In the early nineties, the World Wide Web (WWW) was invented. Nowadays, most people think that the WWW simply consists of all the pretty (or not so pretty) HTML-pages that you can read with your WWW browser. But back then, one of the main intentions behind the design of the WWW was to unify several existing communication protocols.
Then (and even now), information on the Internet was available via a multitude of channels: FTP, HTTP, E-Mail, News, Gopher, and many more. Thanks to the WWW, all these services can now be uniformly addressed via URLs (Uniform Resource Locators). For our problem, we consider a simplified version of the syntax, which is as follows: "://" [":"] ["/"]
The square brackets [] mean that the enclosed string is optional and may or may not appear.
is a string consisting of alphabetic (a-z, A-Z) or numeric (0-9) characters and points (.).
You are to write a program that parses an URL into its components.

Input

The input starts with a line containing a single integer n, the number of URLs in the input. The following n lines contain one URL each, in the format described above. The URLs will consist of at most 60 characters each.

Output

For each URL in the input first print the number of the URL, as shown in the sample output. Then print four lines, stating the protocol, host, port and path specified by the URL. If the port and/or path are not given in the URL, print the string instead.

Sample Input:

```
3
ftp://acm.baylor.edu:1234/pub/staff/mr-p
http://www.informatik.uni-ulm.de/acm
gopher://veryold.edu
```

Sample Output:

```
URL #1
Protocol = ftp
Host     = acm.baylor.edu
Port     = 1234
Path     = pub/staff/mr-p

URL #2
Protocol = http
Host     = www.informatik.uni-ulm.de
Port     =
Path     = acm

URL #3
Protocol = gopher
Host     = veryold.edu
Port     =
Path     =
```

F) Smart File Name Sorting

You have likely tried to sort a number of files in a directory based on their names. As you have noticed, in old basic environments, the file names are sorted in the ASCII-based lexicographic order. The sorting of the alphanumeric ASCII characters is as follows:

0 < 1 < … < 9 < A < B < … < Z < a < b < … < z

Thus, the following file names would be placed in the following order:

A,A0,A01,A02,A1,A10,A2,AA,AB,Aa,Ab,B,B0,a,a0

But, the sorting that we usually would like is the following:

a,A,a0,A0,A01,A1,A02,A2,A10,Aa,AA,Ab,AB,B,B0

Our desired sorting can be formally defined with specifying the way of comparing two file names:

1. If two file names are exactly the same, they are equal. Otherwise, they are not considered to be equal!
2. Any maximal block of consecutive digits in the file name should be considered as a single number. So, a file name is in fact a sequence of letters and numbers.
3. Two unequal file names are compared in two phases. Phase 2 is used only if the order of the two file names could not be distinguished during Phase 1.
4. Phase 1 (soft comparison): The file names are compared lexicographically based on the following rules:
   a. Numbers precede letters (a1<aa).
   b. Numbers with lower values precede numbers with higher values (a2<a10).
   c. Numbers with the same value are not distinguished in this phase.
   d. The letters are compared case-insensitively (in this phase only).
5. Phase 2 (exact/strict comparison): The file names are compared lexicographically based on the following rules:
   a. Numbers with the same value (but with different sequence of digits) are compared lexicographically (01<1<02<2<10).
   b. Lower case of each letter precedes its upper case form (a<A<b<B).

Now, you have to write the “compare” method of our desired sorting algorithm.

Input

Each test case consists of two lines. The first string and the second string appear on the first line and the second line, respectively. Both strings are strings of at most 255 alphanumeric characters. The input terminates with "###" which should not be processed.

Output

For each test case output ‘<’, ‘=’ or ‘>’ (omit the quotes) in one line as described in the following.

- ‘<’: if the first string precedes the second one (in our desired sorting)
- ‘=’: if the two strings are exactly the same
- ‘>’: if the first string succeeds the second one (in our desired sorting)

Sample Input:

```
A
A
A
b
A
a
1
b
Abc
aBD
010
2
010
10
A10
a2
a
aa
A
aa
10c03
10b3
1a2b3d
01a002b3d0
###
```

Sample Output:

```
=
<
>
<
<
>
<
>
<
<
>
<
```

G) Prime Bases

Given any integer base b >= 2, it is well known that every positive integer n can be uniquely represented in base b. That is, we can write

$n = a_0 + a_1*b + a_2*b*b + a_3*b*b*b + ...$

where the coefficients a_0, a_1, a_2, a_3, ... are between 0 and b-1 (inclusive).

What is less well known is that if p_0, p_1, p_2, ... are the first primes (starting from 2, 3, 5, ...), every positive integer n can be represented uniquely in the "mixed" bases as:

$n = a_0 + a_1*p_0 + a_2*p_0*p_1 + a_3*p_0*p_1*p_2 + ...$

where each coefficient a_i is between 0 and p_i-1 (inclusive). Notice that, for example, a_3 is between 0 and p_3-1, even though p_3 may not be needed explicitly to represent the integer n.

Given a positive integer n, you are asked to write n in the representation above. Do not use more primes than it is needed to represent n, and omit all terms in which the coefficient is 0.

Input

Each line of input consists of a single positive 32-bit signed integer. The end of input is indicated by a line containing the integer 0.

Output

For each integer, print the integer, followed by a space, an equal sign, and a space, followed by the mixed base representation of the integer in the format shown below. The terms should be separated by a space, a plus sign, and a space. The output for each integer should appear on its own line.

Sample Input:

```
123
456
123456
0
```

Sample Output:

```
123 = 1 + 1*2 + 4*2*3*5
456 = 1*2*3 + 1*2*3*5 + 2*2*3*5*7
123456 = 1*2*3 + 6*2*3*5 + 4*2*3*5*7 + 1*2*3*5*7*11 + 4*2*3*5*7*11*13
```

H) 2^x mod n = 1

Give a number n, find the minimum x that satisfies 2^x mod n = 1.

Input

One positive integer on each line, the value of n. The end of input is indicated by a line containing the integer 0.

Output

If the minimum x exists, print a line with 2^x mod n = 1.

Print 2^? mod n = 1 otherwise.

You should replace x and n with specific numbers.

Sample Input:

```
2
5
0
```

Sample Output:

```
2^? mod 2 = 1
2^4 mod 5 = 1
```