Problem A

103 Stacking Boxes

Background

Some concepts in Mathematics and Computer Science are simple in one or two dimensions but become more complex when extended to arbitrary dimensions. Consider solving differential equations in several dimensions and analyzing the topology of an n-dimensional hypercube. The former is much more complicated than its one dimensional relative while the latter bears a remarkable resemblance to its "lower-class" cousin.

The Problem

Consider an n-dimensional "box" given by its dimensions. In two dimensions the box (2,3) might represent a box with length 2 units and width 3 units. In three dimensions the box (4,8,9) can represent a box $4 \times 8 \times 9$ (length, width, and height). In 6 dimensions it is, perhaps, unclear what the box (4,5,6,7,8,9) represents; but we can analyze properties of the box such as the sum of its dimensions.

In this problem you will analyze a property of a group of n-dimensional boxes. You are to determine the longest *nesting string* of boxes, that is a sequence of boxes $b_1, b_2, \ldots, b_k$ such that each box b_i nests in box b_{i+1} ($1 \leq i < k$).

A box D = $(d_1, d_2, \ldots, d_n)$ nests in a box E = $(e_1, e_2, \ldots, e_n)$ if there is some rearrangement of the d_i such that when rearranged each dimension is less than the corresponding dimension in box E. This loosely corresponds to turning box D to see if it will fit in box E. However, since any rearrangement suffices, box D can be contorted, not just turned (see examples below).

For example, the box D = (2,6) nests in the box E = (7,3) since D can be rearranged as (6,2) so that each dimension is less than the corresponding dimension in E. The box D = (9,5,7,3) does NOT nest in the box E = (2,10,6,8) since no rearrangement of D results in a box that satisfies the nesting property, but F = (9,5,7,1) does nest in box E since F can be rearranged as (1,9,5,7) which nests in E.

Formally, we define nesting as follows: box D = $(d_1, d_2, \ldots, d_n)$ *nests* in box E = $(e_1, e_2, \ldots, e_n)$ if there is a permutation π of $1 \ldots n$ such that $(d_{\pi(1)}, d_{\pi(2)}, \ldots, d_{\pi(n)})$ "fits" in $(e_1, e_2, \ldots, e_n)$ i.e., if $d_{\pi(i)} < e_i$ for all $1 \leq i \leq n$.

The Input

The input consists of a series of box sequences. Each box sequence begins with a line consisting of the the number of boxes k in the sequence followed by the dimensionality of the boxes, n (on the same line.)

This line is followed by k lines, one line per box with the n measurements of each box on one line separated by one or more spaces. The i^{th} line in the sequence ($1 \leq i \leq k$) gives the measurements for the i^{th} box.

There may be several box sequences in the input file. Your program should process all of them and determine, for each sequence, which of the k boxes determine the longest nesting string and the length of that nesting string (the number of boxes in the string).

In this problem the maximum dimensionality is 10 and the minimum dimensionality is 1. The maximum number of boxes in a sequence is 30.

The Output

For each box sequence in the input file, output the length of the longest nesting string on one line followed on the next line by a list of the boxes that comprise this string in order. The "smallest" or "innermost" box of the nesting string should be listed first, the next box (if there is one) should be listed second, etc.

The boxes should be numbered according to the order in which they appeared in the input file (first box is box 1, etc.).

If there is more than one longest nesting string then any one of them can be output.

Sample Input

```
5 2
3 7
8 10
5 2
9 11
21 18
8 6
5 2 20 1 30 10
23 15 7 9 11 3
40 50 34 24 14 4
9 10 11 12 13 14
31 4 18 8 27 17
44 32 13 19 41 19
1 2 3 4 5 6
80 37 47 18 21 9
```

Sample Output

```
5
3 1 2 4 5
4
7 2 5 6
```

Problem B

160 Factors and Factorials

The factorial of a number N (written $N!$) is defined as the product of all the integers from 1 to N. It is often defined recursively as follows:

$$1! = 1$$

$$N! = N * (N-1)!$$

Factorials grow very rapidly—5! = 120, 10! = 3,628,800. One way of specifying such large numbers is by specifying the number of times each prime number occurs in it, thus 825 could be specified as (0 1 2 0 1) meaning no twos, 1 three, 2 fives, no sevens and 1 eleven.

Write a program that will read in a number N ($2 \leq N \leq 100$) and write out its factorial in terms of the numbers of the primes it contains.

Input

Input will consist of a series of lines, each line containing a single integer N. The file will be terminated by a line consisting of a single 0.

Output

Output will consist of a series of blocks of lines, one block for each line of the input. Each block will start with the number N, right justified in a field of width 3, and the chracters '`!`', space, and '`=`'. This will be followed by a list of the number of times each prime number occurs in $N!$.

These should be right justified in fields of width 3 and each line (except the last of a block, which may be shorter) should contain fifteen numbers. Any lines after the first should be indented. Follow the layout of the example shown below exactly.

Sample input

```
5
53
0
```

Sample output

```
  5! =  3  1  1
 53! = 49 23 12  8  4  4  3  2  2  1  1  1  1  1  1
        1
```

Problem C

196 Spreadsheet

In 1979, Dan Bricklin and Bob Frankston wrote VisiCalc, the first spreadsheet application. It became a huge success and, at that time, was the killer application for the Apple II computers. Today, spreadsheets are found on most desktop computers.

The idea behind spreadsheets is very simple, though powerful. A spreadsheet consists of a table where each cell contains either a number or a formula. A formula can compute an expression that depends on the values of other cells. Text and graphics can be added for presentation purposes.

You are to write a very simple spreadsheet application. Your program should accept several spreadsheets. Each cell of the spreadsheet contains either a numeric value (integers only) or a formula, which only support sums. After having computed the values of all formulas, your program should output the resulting spreadsheet where all formulas have been replaced by their value.

A1	B1	C1	D1	E1	F1	...
A2	B2	C2	D2	E2	F2	...
A3	B3	C3	D3	E3	F3	...
A4	B4	C4	D4	E4	F4	...
A5	B5	C5	D5	E5	F5	...
A6	B6	C6	D6	E6	F6	...
...	...	...	...	...	...	...

Figure 1: Naming of the top left cells

Input

The first line of the input file contains the number of spreadsheets to follow. A spreadsheet starts with a line consisting of two integer numbers, separated by a space, giving the number of columns and rows. The following lines of the spreadsheet each contain a row. A row consists of the cells of that row, separated by a single space.

A cell consists either of a numeric integer value or of a formula. A formula starts with an equal sign (`=`). After that, one or more cell names follow, separated by plus signs (`+`). The value of such a formula is the sum of all values found in the referenced cells. These cells may again contain a formula. There are no spaces within a formula.

You may safely assume that there are no cyclic dependencies between cells. So each spreadsheet can be fully computed.

The name of a cell consists of one to three letters for the column followed by a number between 1 and 999 (including) for the row. The letters for the column form the following series: A, B, C, ..., Z, AA, AB, AC, ..., AZ, BA, ..., BZ, CA, ..., ZZ, AAA, AAB, ..., AAZ, ABA, ..., ABZ, ACA, ..., ZZZ. These letters correspond to the number from 1 to 18278. The top left cell has the name `A1`. See figure 1.

Output

The output of your program should have the same format as the input, except that the number of spreadsheets and the number of columns and rows are not repeated. Furthermore, all formulas should be replaced by their value.

Sample Input

```
1
4 3
10 34 37 =A1+B1+C1
40 17 34 =A2+B2+C2
=A1+A2 =B1+B2 =C1+C2 =D1+D2
```

Sample Output

```
10 34 37 81
40 17 34 91
50 51 71 172
```

Problem D

195 Anagram

You are to write a program that has to generate all possible words from a given set of letters.

Example: Given the word "`abc`", your program should - by exploring all different combination of the three letters - output the words "`abc`", "`acb`", "`bac`", "`bca`", "`cab`" and "`cba`".

In the word taken from the input file, some letters may appear more than once. For a given word, your program should not produce the same word more than once, and the words should be output in alphabetically ascending order.

Input

The input file consists of several words. The first line contains a number giving the number of words to follow. Each following line contains one word. A word consists of uppercase or lowercase letters from A to Z. Uppercase and lowercase letters are to be considered different.

Output

For each word in the input file, the output file should contain all different words that can be generated with the letters of the given word. The words generated from the same input word should be output in alphabetically ascending order.

Sample Input

```
2
abc
acba
```

Sample Output

```
abc
acb
bac
bca
cab
cba
aabc
aacb
abac
abca
acab
acba
baac
baca
bcaa
caab
caba
cbaa
```

Problem E

Problem H
Series-Parallel Networks

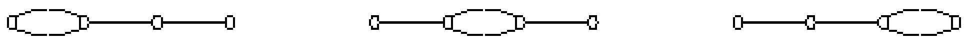
Input: standard input
Output: standard output
Time Limit: 5 seconds
Memory Limit: 32 MB

In this problem you are expected to count **two-terminal series-parallel networks**. These are electric networks considered topologically or geometrically, that is, without the electrical properties of the elements connected. One of the two terminals can be considered as the **source** and the other as the **sink**.

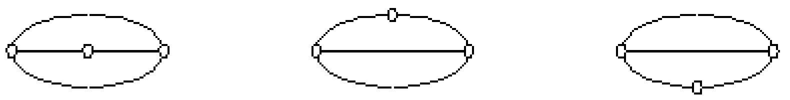
A two-terminal network will be considered series-parallel if it can be obtained iteratively in the following way:

- A single edge is two-terminal series-parallel.
- If $\mathbf{G_1}$ and $\mathbf{G_2}$ are two-terminal series-parallel, so is the network obtained by identifying the sources and sinks, respectively (parallel composition).
- If $\mathbf{G_1}$ and $\mathbf{G_2}$ are two-terminal series-parallel, so is the network obtained by identifying the sink of $\mathbf{G_1}$ with the source of $\mathbf{G_2}$ (series composition).

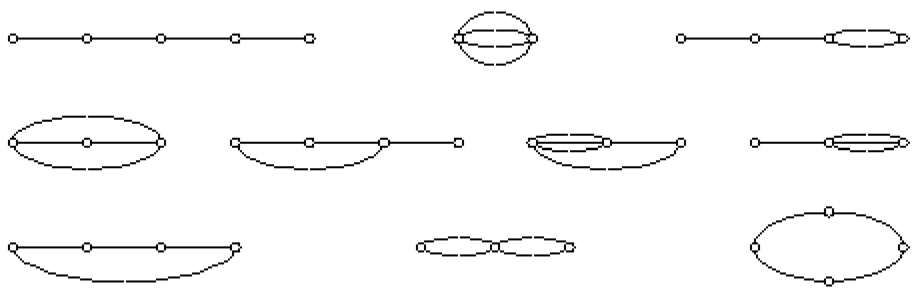
Note here that in a series-parallel network two nodes can be connected by multiple edges. Moreover, networks are regarded as equivalent, not only topologically, but also when interchange of elements in series brings them into congruence; otherwise stated, series interchange is an equivalence operation. For example, the following three networks are equivalent:



Similarly, parallel interchange is also an equivalence operation. For example, the following three networks are also equivalent:



Now, given a number **N**, you are expected to count the number of two-terminal series parallel networks containing exactly **N** edges. For example, for **N = 4**, there are exactly **10** series-parallel networks as shown below:



Input

Each line of the input file contains an integer **N (1 ≤ N ≤ 30)** specifying the number of edges in the network.

A line containing a zero for **N** terminates the input and this input need not be considered.

Output

For each **N** in the input file print a line containing the number of two-terminal series-parallel networks that can be obtained using exactly **N** edges.

Sample Input

```
1
4
15
0
```

Sample Output

```
1
10
1399068
```

(World Final Warm-up Contest, Problem Setter: Rezaul Alam Chowdhury)

Problem F

Problem A: Railway

Problem

Railway is a broken line of *N* segments. The problem is to find such a position for the railway station that the distance from it to the given point *M* is the minimal.

Input

The input will consist of several input blocks. Each input block begins with two lines with coordinates *Xm* and *Ym* of the point *M*. In the third line there is *N* - the number of broken line segments. The next *2N+2* lines contain the *X* and the *Y* coordinates of consecutive broken line corners.

The input is terminated by <EOF>.

Output

For each input block there should be two output lines. The first one contains the first coordinate of the station position, the second one contains the second coordinate. Coordinates are the floating-point values with four digits after decimal point.

Sample Input

```
6
-3
3
0
1
5
5
9
-5
15
3
0
0
1
1
0
2
0
```

Sample Output

```
7.8966
-2.2414
1.0000
```

```
0.0000
```

Olga Zaverach, 2002

Problem G

Problem H
{sum+=i++} to Reach N

Input: standard input
Output: standard output
Memory Limit: 32 MB

All the positive numbers can be expressed as a sum of one, two or more consecutive positive integers. For example **9** can be expressed in three such ways, **2+3+4**, **4+5** or **9**. Given an integer less than **(9*10^14+1)** or **(9E14 + 1)** or **($9*10^{14}$ +1)** you will have to determine in how many ways that number can be expressed as summation of consecutive numbers.

Input

The input file contains less than **1100** lines of input. Each line contains a single integer **N** **(0<=N<= 9E14)**. Input is terminated by end of file.

Output

For each line of input produce one line of output. This line contains an integer which tells in how many ways **N** can be expressed as summation of consecutive integers.

Sample Input

```
9
11
12
```

Sample Output

```
3
2
2
```

(Math Lovers' Contest, Problem Setter: Shahriar Manzoor)