1 The Teacher's Side of Math

Description

One of the tasks students routinely carry out in their mathematics classes is to solve a polynomial equation. It is, given a polynomial, say $X^2 - 4X + 1$, to find its roots $(2 \pm \sqrt{3})$.

If the students' task is to find the roots of a given polynomial, the teacher's task is then to find a polynomial that has a given root. Ms. Galsone is an enthusiastic mathematics teacher who is bored with finding solutions of quadratic equations that are as simple as $a + b\sqrt{c}$. She wanted to make higher-degree equations whose solutions are a little more complicated. As usual in problems in mathematics classes, she wants to maintain all coefficients to be integers and keep the degree of the polynomial as small as possible (provided it has the specified root). Please help her by writing a program that carries out the task of the teacher's side.

You are given a number t of the form:

$$t = \sqrt[m]{a} + \sqrt[n]{b}$$

where a and b are distinct prime numbers, and m and n are integers greater than 1.

In this problem, you are asked to find t's *minimal polynomial on integers*, which is the polynomial $F(X) = a_d X^d + a_{d-1} X^{d-1} + \ldots + a_1 X + a_0$ satisfying the following conditions.

1. Coefficients $a_0, \ldots, a_d$ are integers and $a_d > 0$.
2. $F(t) = 0$.
3. The degree d is minimum among polynomials satisfying the above two conditions.
4. $F(X)$ is primitive. That is, coefficients $a_0, \ldots, a_d$ have no common divisors greater than one.

For example, the minimum polynomial of $\sqrt{3} + \sqrt{2}$ on integers is $F(X) = X^4 - 10X^2 + 1$. verifying $F(t) = 0$ is as simple as the following $(\alpha = \sqrt{3}, \beta = \sqrt{2})$.

$$\begin{aligned} F(t) &= (\alpha + \beta)^4 - 10(\alpha + \beta)^2 + 1 \\ &= (\alpha^4 + 4\alpha^3\beta + 6\alpha^2\beta^2 + 4\alpha\beta^3 + \beta^4) - 10(\alpha^2 + 2\alpha\beta + \beta^2) + 1 \\ &= 9 + 12\alpha\beta + 36 + 8\alpha\beta + 4 - 10(3 + 2\alpha\beta + 2) + 1 \\ &= (9 + 36 + 4 - 50 + 1) + (12 + 8 - 20)\alpha\beta \\ &= 0 \end{aligned}$$

Verifying that the degree of $F(t)$ is in fact minimum is a bit more difficult. Fortunately, under the condition given in this problem, which is that a and b are distinct prime numbers and m and n greater than one, the degree of the minimal polynomial is always mn. Moreover, it is always *monic*. That is, the coefficient of its highest-order term (a_d) is one.

Input

The input consists of multiple datasets, each in the following format.

$$a\ m\ b\ n$$

This line represents $\sqrt[m]{a} + \sqrt[n]{b}$. The last dataset is followed by a single line consisting of four zeros. Numbers in a single line are separated by a single space.

Every dataset satisfies the following conditions.

1. $\sqrt[m]{a} + \sqrt[n]{b} \leq 4$
2. $mn \leq 20$
3. The coefficients of the answer $a_0, \ldots, a_d$ are between $(2^{31} + 1)$ and $(2^{31} - 1)$, inclusive.

Output

For each dataset, output the coefficients of its minimal polynomial on integers $F(X) = a_d X^d + a_{d-1} X^{d-1} + \ldots + a_1 X + a_0$, in the following format.

$$a_d \ a_{d-1} \ \ldots \ a_1 \ a_0$$

Non-negative integers must be printed without a sign (+ or). Numbers in a single line must be separated by a single space and no other characters or extra spaces may appear in the output.

Example

Sample Input

```
3 2 2 2
3 2 2 3
2 2 3 4
31 4 2 3
3 2 2 7
0 0 0 0
```

Sample Output

```
1 0 -10 0 1
1 0 -9 -4 27 -36 -23
1 0 -8 0 18 0 -104 0 1
1 0 0 -8 -93 0 24 -2976 2883 -32 -3720 -23064 -29775
1 0 -21 0 189 0 -945 -4 2835 -252 -5103 -1260 5103 -756 -2183
```

2 Suffix reconstruction

Description

Given a text $s[1 \dots n]$ of length n, we create its suffix array by taking all its suffixes:

$$s[1 \dots n], s[2 \dots n], \dots, s[n \dots n]$$

and sorting them lexicographically. As a result we get a sorted list of suffixes:

$$s[p(1) \dots n], s[p(2) \dots n], \dots, s[p(n) \dots n]$$

and call the sequence $p(1), p(2), \dots, p(n)$ the suffix array of $s[1 \dots n]$. For example, if $s = abbaabab$, the sorted list of all suffixes becomes:

$$aabab, ab, abab, abbaabab, b, baabab, bab, bbaabab$$

and the suffix array is 4, 7, 5, 1, 8, 3, 6, 2.

It turns out that it is possible to construct this array in a linear time. Your task will be completely different, though: given $p(1), p(2), p(3), \dots, p(n)$ you should check if there exist at least one text consisting of lowercase letters of the English alphabet for which this sequence is the suffix array. If so, output any such text. Otherwise output -1.

Input

The input contains several descriptions of suffix arrays. The first line contains the number of descriptions $t (t \leq 100)$. Each description begins with a line containing the length of both the text and the array $n (1 \leq n \leq 500000)$. Next line contains integers $p(1), p(2), \dots, p(n)$. You may assume that $1 \leq p(i) \leq n$ and no value of $p(i)$ occurs twice. Total size of the input will not exceed 50MB.

Output

For each test case output any text resulting in the given suffix array. In case there is no such text consisting of lowercase letters of the English alphabet, output -1.

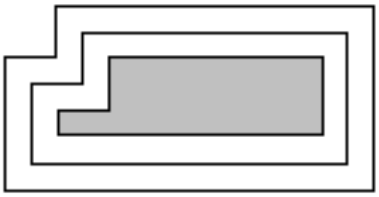
Example

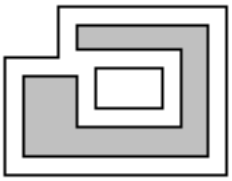
Input	Output
5 2 1 2 2 2 1 3 2 3 1 6 3 4 5 1 2 6 14 3 10 2 12 14 5 13 4 1 8 6 11 7 9 7 5 1 7 4 3 2 6	ab aa bab suffix reconstruction issofun

3 Bordering on Madness

Description

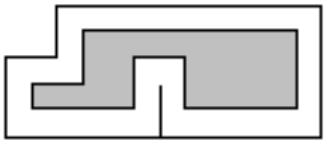
Bob Roberts owns a design business which creates custom artwork for various corporations. One technique that his company likes to use is to take a simple rectilinear figure (a figure where all sides meet at 90 or 270 degrees and which contains no holes) and draw one or more rectilinear borders around them. Each of these borders is drawn so that it is a set distance d away from the previously drawn border (or the original figure if it is the first border) and then the new area outlined by each border is painted a unique color. Some examples are shown below (without the coloring of the borders).

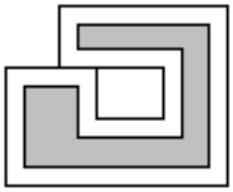




The example on the left shows a simple rectilinear figure (grey) with two borders drawn around it. The one on the right is a more complicated figure; note that the border may become disconnected.

These pieces of art can get quite large, so Bob would like a program which can draw prototypes of the finished pieces in order to judge how aesthetically pleasing they are (and how much money they will cost to build). To simplify things, Bob never starts with a figure that results in a border where 2 horizontal (or vertical) sections intersect, even at a point. This disallows such cases as those shown below:





Input

Input will consist of multiple test cases. The first line of the input file will contain a single integer indicating the number of test cases. Each test case will consist of two or more lines. The first will contain three positive integers n, m and d indicating the number of sides of the rectlinear figure, the number of borders to draw, and the distance between each border, where $n \leq 100$ and $m \leq 20$. The remaining lines will contain the n vertices of the figure, each represented by two positive integers indicating the x and y coordinates. The vertices will be listed in clockwise order starting with the vertex with the largest y value and (among those vertices) the smallest x value.

Output

For each test case, output three lines: the first will list the case number (as shown in the examples), the second will contain m integers indicating the length of each border, and the third will contain m integers indicating the additional area contributed to the artwork by each border. Both of these sets of numbers should be listed in order, starting from the border nearest the original figure. Lines two and three should be indented two spaces and labeled as shown in the examples. Separate test cases with a blank line.

Example

Sample Input	Sample Output

2
6 2 10
20 30 100 30 100 0 0 0 0 10 20 10
10 1 7
20 50 70 50 70 0 0 0 0 30
20 30 20 10 60 10 60 40 20 40
	Case 1:
	  Perimeters: 340 420
	  Areas: 3000 3800
	Case 2:
	  Perimeters: 380
	  Areas: 2660

4 Give Me an E

Description

Everyone knows that the letter "E" is the most frequent letter in the English language. In fact, there are one hundred sixteen E's on this very page ... no, make that one hundred twenty one. Indeed, when spelling out integers it is interesting to see which ones do NOT use the letter "E". For example 6030 (six thousand thirty) doesn't. Nor does 4002064 (four million two thousand sixty four).

It turns out that 6030 is the 64^{th} positive integer that does not use an "E" when spelled out and 4002064 is the 838^{th} such number. Your job is to find the n-th such number.

Note: 1,001,001,001,001,001,001,001,001,000 is "one octillion, one septillion, one sextillion, one quintillion, one quadrillion, one trillion, one billion, one million, one thousand". (Whew!)

Input

The input file will consist of multiple test cases. Each input case will consist of one positive integer n (less than 2^{31}) on a line. A 0 indicates end-of-input. (There will be no commas in the input.)

Output

For each input n you will print, with appropriate commas, the n-th positive integer whose spelling does not use an "E". You may assume that all answers are less than 10^{28}.

Example

Sample Input	Sample Output
1 10 838 0	2 44 4,002,064

5 Slim Span

Description

Given an undirected weighted graph G, you should find one of spanning trees specified as follows.

The graph G is an ordered pair (V, E), where V is a set of vertices $\{v_1, v_2, \ldots, v_n\}$ and E is a set of undirected edges $\{e_1, e_2, \ldots, e_m\}$. Each edge $e \in E$ has its weight $w(e)$.

A spanning tree T is a tree (a connected subgraph without cycles) which connects all the n vertices with $n-1$ edges. The slimness of a spanning tree T is defined as the difference between the largest weight and the smallest weight among the $n-1$ edges of T.

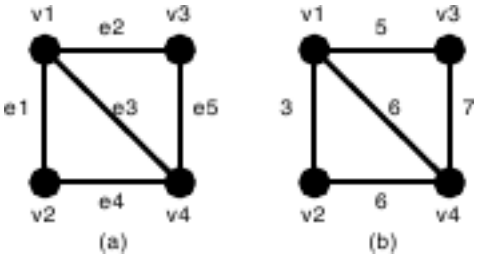


Figure 1: A Graph G and the weights of the edges

For example, a graph G in Figure 1(a) has four vertices $\{v_1, v_2, v_3, v_4\}$ and five undirected edges $\{e_1, e_2, e_3, e_4, e_5\}$. The weights of the edges are $w(e_1) = 3$, $w(e_2) = 5$, $w(e_3) = 6$, $w(e_4) = 6$, $w(e_5) = 7$ as shown in Figure 1(b).

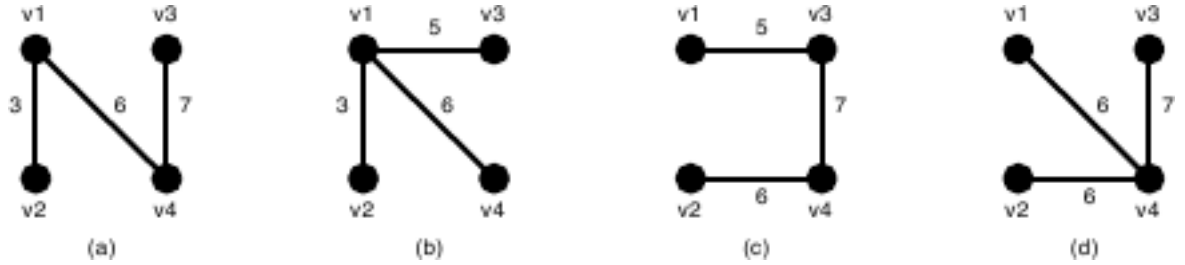


Figure 2: Example of spanning trees of G

There are several spanning trees for G. Four of them are depicted in Figure 2(a)(d). The spanning tree Ta in Figure 2(a) has three edges whose weights are 3, 6 and 7. The largest weight is 7 and the smallest weight is 3 so that the slimness of the tree T_a is 4. The slimnesses of spanning trees T_b , T_c and T_d shown in Figure 2(b), (c) and (d) are 3, 2 and 1, respectively. You can easily see the slimness of any other spanning tree is greater than or equal to 1, thus the spanning tree Td in Figure 2 (d) is one of the slimmest spanning trees whose slimness is 1.

Your job is to write a program that computes the smallest paragraphslimness.

Input

The input consists of multiple datasets, followed by a line containing two zeros separated by a space. Each dataset has the following format.

$$\begin{array}{lll} n & m & \\ a_1 & b_1 & w_1 \\ & \vdots & \\ a_m & b_m & w_m \end{array}$$

Every input item in a dataset is a non-negative integer. Items in a line are separated by a space.

n is the number of the vertices and m the number of the edges. You can assume $2 \leq n \leq 100$ and $0 \leq m \leq \frac{n(n-1)}{2}$. a_k and b_k $(k = 1, \ldots, m)$ are positive integers less than or equal to n, which represent the two vertices v_{a_k} and v_{b_k} connected by the kth edge e_k. w_k is a positive integer less than or equal to 10000, which indicates the weight of e_k . You can assume that the graph $G = (V, E)$ is simple, that is, there are no self-loops that connect the same vertex) nor parallel edges (that are two or more edges whose both ends are the same two vertices).

Output

For each dataset, if the graph has spanning trees, the smallest slimness among them should be printed. Otherwise, 1 should be printed. An output should not contain extra characters.

Example

Sample Input	Sample Output
4 5 1 2 3 1 3 5 1 4 6 2 4 6 3 4 7 4 6 1 2 10 1 3 100 1 4 90 2 3 20 2 4 80 3 4 40 2 1 1 2 1 3 0 3 1 1 2 1 3 3 1 2 2 2 3 5 1 3 6 5 10 1 2 110 1 3 120 1 4 130 1 5 120 2 3 110 2 4 120 2 5 130 3 4 120 3 5 110 4 5 120 5 10 1 2 9384 1 3 887 1 4 2778 1 5 6916 2 3 7794 2 4 8336 2 5 5387 3 4 493 3 5 6650 4 5 1422 5 8 1 2 1 2 3 100 3 4 100 4 5 100 1 5 50 2 5 50 3 5 50 4 1 150 0 0	1 20 0 -1 -1 1 0 1686 50

6 The Two Note Rag

Description

Since most computers are binary machines, both powers of two and problems that involve only two values are important to computer scientists. The following problem has to do with powers of two and the digits 1 and 2.

Some powers of two as decimal values, such as $2^9 = 512$ and $2^{89} = 618,970,019,642,690,137,449,562,112$ end in a string of digits consisting only of 1's and 2's (12 for 2^9 and 2112 for 2^{89}). In fact, it can be proved that:

> *For every integer R, there exists a power of 2 such that 2^K uses only the digits 1 and 2 in its last R digits.*

This is shown a bit more clearly in the following table:

R	Smallest K	2^K
1	1	2
2	9	512
3	89	...112
4	89	...2112
5	589	...22112
6	3089	...122112
...	...	...

Your job is to write a program that will determine, for given R, the smallest K such that 2^K ends in a string of R digits containing only 1's and 2's.

Input

The first line of the input contains a single decimal integer, $N, 1 \leq N \leq 50$, the number of problem data sets to follow. Each data set consists of a single integer $R, 1 \leq R \leq 20$, for which we want a power of 2 ending in a string of R 1's and 2's.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space, the input value R, another space, and the smallest value K for which 2^K ends in a string of R 1's and 2's.

Example

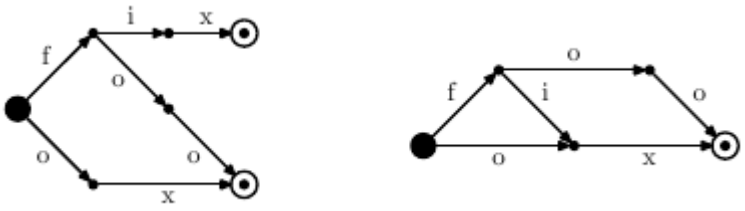
Sample Input	Sample Output
6 1 2 4 5 7 15	1 1 1 2 2 9 3 4 89 4 5 589 5 7 3089 6 15 11687815589

7 Language Recognition

Description

Deterministic Final-State Automaton (DFA) is a directed multigraph whose vertices are called states and edges are called transitions. Each DFA transition is labeled with a single letter. Moreover, for each state s and each letter l there is at most one transition that leaves s and is labeled with l. DFA has a single starting state and a subset of final states. DFA defines a language of all words that can be constructed by writing down the letters on a path from the starting state to some final state.

Given a language with a finite set of words it is always possible to construct a DFA that defines this language. The picture on the left shows such DFA for the language cosisting of three words: `fix`, `foo`, `ox`. However, this DFA has 7 states, which is not optimal. The DFA on the right defines the same language with just 5 states.



Your task is to find the minimum number of states in a DFA that defines the given language.

Input

The first line of the input file contains a single integer number $n (1 \le n \le 5000)$ the number of words in the language. It is followed by n lines with a word on each line. Each word consists of 1 to 30 lowercase Latin letters from "`a`" to "`z`". All words in the input file are different.

Output

Write to the output file a single integer number the minimal number of states in a DFA that defines the language from the input file.

Example

Sample Input	Sample Output
3 fix foo ox 4 a ab ac ad	5 3

8 Lingo

Description

You are going to participate in the television show 'lingo'. You are very confident that you will make the finals of the show. This is not only because you are well prepared, but also because you managed to find a way to use your pda unnoticed during the contest, and you have already written a program to help you with the bonuswords.

In the finals of the show, you first solve as many lingo words as possible within the allowed time. This determines how many balls you may take afterwards. The more balls you may take, the higher your probability is of winning the finals. But it is not easy to see what this probability is. Write a program to help you with this.

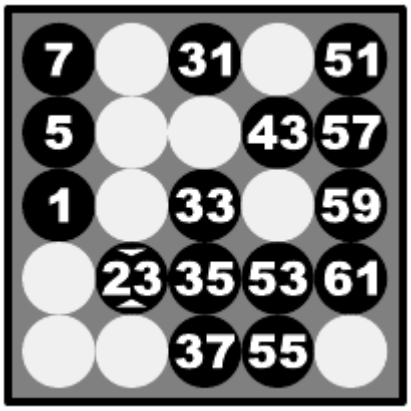


Figure 3: Grid of first sample case

For those who don't know the game of lingo, here follows a description of the last part of the finals, where you take the balls. You are given a square grid. Some squares in this grid are covered and the other squares contain numbers. A hopper in front of you contains numbered balls; there is exactly one ball for each numbered grid square. You take a ball at random (without replacement) from this hopper for each lingo word you solved in the first part of the finals. When you take a ball, the corresponding square in the grid becomes covered. You win the finals if an entire row, column or diagonal consist of only covered squares.

Input

On the first line an integer $t(1 \leq t \leq 100)$: the number of test cases. Then for each test case:

- One line with the integers $n(1 \leq n \leq 8)$ and $k(0 \leq k)$, where n is the size of the lingo grid and k is the number of words you solved in the first part of the finals.
- n lines, with on each line exactly n characters. Each character will be either '*' or '.', representing a covered square and a numbered square respectively.

There will be at least k numbered squares on the board, and there is no row, column or diagonal covered yet.

Output

For each test case:

- One line with the percentage of getting lingo with either an absolute or a relative error of at most 10^{-6}.

Example

Sample Input	Sample Output
1 5 7 .*.*. .**.. .*.*. *.... **..*	82.703962704