Problem A. Breaking Hashing

Input file:	`breaking-hashing.in`
Output file:	`breaking-hashing.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Your solutions fail on corner cases?
Built-in quick sort suddenly works in quadratic time?
You struggle hard against precision losses in geometric problems?
Your solutions pass local stress testing but fail on the judges' tests?
The bug turned out to be in the library routine and not in your solution?

You want to know who is behind all of that?

Today you have a unique opportunity to join a secret organization:
the Order of Cunning Beavers! Members of this organization make approximately 146% more successful hacks than uninitiated folks, and in the problems authored by them, the average number of missed test cases is significantly less. To apply for membership, one has to pass the entrance examination: solve the problem below.

Hurry! Only the first few will be able to apply!

In this problem, it is required to find a collision for polynomial hashing of strings consisting of lowercase English letters.

A *polynomial hash* of a string has two parameters: multiplier p and modulus q. For empty string ε, the value of hash function $h(\varepsilon) = 0$, and for any string S and character c it is defined recursively by the following formula: $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Here, $\text{code}(c)$ is the ASCII code of character c. Recall that codes of lowercase English letters are consecutive: code('`a`') = 97, code('`b`') = 98, ..., code('`z`') = 122. We can also obtain a general formula: if $S = s_1 s_2 \dots s_n$, we have $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

Given the parameters p and q, find two different non-empty strings A and B such that $h(A) = h(B)$.

Input

The first line of input contains two integers p and q separated by a space: the parameters of the hash function ($0 < p < q < 2 \cdot 10^{18}$).

Output

On the first two lines, output two different non-empty strings A and B such that $h(A) = h(B)$. The strings should consist entirely of lowercase English letters (ASCII codes 97–122) and have lengths from 1 to 100 000 characters. Note that lenghts of the strings are not required to be equal. In case of multiple answers, output any one of them.

Example

breaking-hashing.in	breaking-hashing.out
31 47	aa bq

Explanation

In the example, $h(A) = (97 \cdot 31 + 97) \bmod 47 = 3104 \bmod 47 = 2$ and
$h(B) = (98 \cdot 31 + 113) \bmod 47 = 3151 \bmod 47 = 2$.

Problem B. Searching for the Rectangle

Input file:	`find-rect.in`
Output file:	`find-rect.out`
Time limit:	0.5 seconds (*0.5 seconds for Java*)
Memory limit:	256 mebibytes

A grid $W \times H$ is given. Each cell is either black or white. Some time ago, the grid was wholly white. Then, Vasya came and filled a rectangle of size $w \times h$ with black ($w, h \geq 15$). A bit later, Noises came, and each cell changed its color to the opposite one (black to white, white to black) with probability 0.05 independently of other cells.

You are given the resulting grid $W \times H$. Your task is to recover the rectangle drawn by Vasya.

To break ambiguous situations, let us formalize the task. We will denote coordinates of a cell by a pair (x, y) ($1 \leq x \leq W$, $1 \leq y \leq H$). Let your answer be rectangle r:

$$r = [x_1 \ldots x_2] \times [y_1 \ldots y_2].$$

Furthermore, let $dx = \max(15, x_2 - x_1 + 1)$ and $dy = \max(15, y_2 - y_1 + 1)$. Consider the rectangle $q(r)$:

$$q(r) = [x_1 - dx \ldots x_2 + dx] \times [y_1 - dy \ldots y_2 + dy].$$

Let $\text{black}(r)$ be the number of black cells in rectangle r and $\text{area}(r)$ be the total number of cells in that rectangle. Then your task is to find a rectangle r with maximal possible $\text{Score}(r)$ defined as follows:

$$\text{Score}(r) = \frac{\text{black}(r)}{\text{area}(r) + (\text{black}(q(r)) - \text{black}(r))}.$$

Input

On the first line of input, there are two integers W and H ($15 \leq W, H \leq 1000$). Each of the next H lines contains exactly W zeroes and ones. Zeroes correspond to white cells, ones to black cells.

It is guaranteed that each test is generated by the algorithm described above: a white rectangle is taken, and a black rectangle is then drawn inside (the size of the black one is at least 15×15). After that, the generator program adds Noises (with probability 0.05 independently in each cell).

Output

Output four integers x_1, x_2, y_1 and y_2: the coordinates of two opposite corners of Vasya's rectangle ($1 \leq x_1 \leq x_2 \leq W$, $1 \leq y_1 \leq y_2 \leq H$). If there are several rectangles with maximum Score, you may output any of them.

Example

find-rect.in	find-rect.out
22 18 00000000000000000001 00100111111111111110 00100111111110111110 00000011111111111110 00100111111111111110 00000011111111011110 00000111111011111110 00000111111111111110 01000111111110111110 00100111111011111110 00000111111111111110 00000111111111011110 00000111111111110111110 00100111110111111110110 00000111101111111110 00000111111111111011110 01000011111111111101110 00001111111111111111110	6 21 2 18

Explanation

In this example, the rectangle in the answer is $r = [6...21] \times [2...18]$.
By the first coordinate, $x_1 = 6$, $x_2 = 21$, $dx = \max(15, 21 - 6 + 1) = 16$.
By the second coordinate, $y_1 = 2$, $y_2 = 18$, $dy = \max(15, 18 - 2 + 1) = 17$.
Let us now find the rectangle $q(r)$:

$$q(r) = [\max(1, x_1 - dx) \dots \min(W, x_2 + dx)] \times [\max(1, y_1 - dy) \dots \min(H, y_2 + dy)] = [1...22] \times [1...18].$$

Now let us compute Score: $\text{black}(r) = 257$, $\text{black}(q(r)) = 266$, $\text{area}(r) = 272$,

$$\text{Score}(r) = \frac{257}{272 + (266 - 257)} = \frac{257}{281} = 0.9145907\dots$$

Problem C. Grasshopper

Input file:	grasshopper.in
Output file:	grasshopper.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

The Grasshopper lives on a one-dimensional meadow which is a set of points on a coordinate axis: the meadow consists of all points with integer coordinates from -10^{100} to $+10^{100}$, inclusive.

The Grasshopper has three pairs of legs. If the Grasshopper is located at point x, the first pair of legs allows him to jump to point $x + a$, the second pair to point $x + b$, and the third one to point $x + c$. An additional restriction is that the Grasshopper can not jump outside the meadow.

The Grasshopper is an optimist, so he always looks on the side where the coordinates of points increase. So, if the Grasshopper can jump from x to $x + t$, it does not automatically imply that he can also jump from x to $x - t$.

Initially, the Grasshopper is located at point 0. In order to reach some point of the meadow, he can jump any non-negative number of times using each of his three pairs of legs. The jumps by the means of different pairs of legs can occur in any order.

Calculate the approximate portion of points that can be reached by the Grasshopper, that is, the ratio of the number of points he can reach to the total number of points.

Input

The first line of input contains three integers a, b and c separated by spaces: the parameters of the three pairs of Grasshopper's legs ($-10^6 \le a, b, c \le 10^6$).

Output

On the first line, print one real number: the approximate portion of points that can be reached by the Grasshopper. The answer is considered correct if it differs from the exact answer by no more than 10^{-9} in absolute value.

Examples

grasshopper.in	grasshopper.out
2 2 2	0.25
0 -3 2	1.0

Explanations

In the first example, each of the three pairs of legs allows the Grasshopper to reach the point $x + 2$ from any point x. So, he can reach any point with noonegative even coordinate and can not reach any other point. The portion of such points is

$$\frac{5 \cdot 10^{99} + 1}{2 \cdot 10^{100} + 1} \approx 0.25.$$

In the second example, the Grasshopper can move from x to $x - 1$ jumping by $+2$ and -3 in some order. Moreover, he can move from x to $x + 1$ jumping by $+2$, $+2$ and -3 in some order. By repeating one of these sequences a certain number of times, the Grasshopper can reach every point of the meadow.

Problem D. Mondays

Input file:	monday.in
Output file:	monday.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Vasya does not like Mondays at all. He made up a list of n dates: the Mondays which should be cancelled. Each date in the list has the form "yyyy/mm/dd": year, month and day. Years in his list are between 1600 and 9999, inclusive, and month and day always consist of two decimal digits, so they are padded with a zero if needed. Vasya uses the Gregorian calendar.

Then Vasya ciphered the list using a simple substitution cipher: he substituted each digit by a letter from "a" to "j". The same digits correspond to the same letters, different digits to different letters.

Unfortunately, Vasya forgot which letter corresponds to which digit. Help him restore the original list.

Here is a brief definion of the Gregorian calendar. In the Gregorian calendar, there are twelve months in each year. Years can be leap or not leap. Every year that is exactly divisible by 4 and not divisible by 100 is a leap year; the years that are exactly divisible by 400 are also leap years. All the other years are not leap. The length of the second month is 29 days if the year is leap and 28 days otherwise. The length of the first, third, fifth, seventh, eighth, tenth and twelfth months is 31 days. All the other months' length is 30 days. It is known that 2012/12/17 is a Monday.

Input

The first line of input contains an integer n ($1 \le n \le 100\,000$). The next n lines contain ciphered dates in the format "yyyy/mm/dd". In this list, digits are substituted by lowercase English letters from "a" to "j".

Output

Print a string consisting of ten distinct digits: the first digit corresponds to the letter "a", the second one to the letter "b" and so on. If there is more than one correspondence such that all dates in the list are correct, print the string which is lexicographically the least possible. If there is no such correspondence, print "IMPOSSIBLE" instead.

Examples

monday.in	monday.out
1 cabc/bc/bh	0123456789
3 cabc/bc/bh cabc/bc/bg cabc/bc/bf	IMPOSSIBLE

Explanations

In the first example, the correspondence a $\leftrightarrow$ 0, b $\leftrightarrow$ 1, ..., j $\leftrightarrow$ 9 ensures that "2012/12/17" is a correct date corresponding to a Monday: year is between 1600 and 9999, month between 1 and 12, and day in that month between 1 and 31. So, the list is correctly restored. As the resulting string "0123456789" is lexicographically the least possible, it must be printed as the answer.

In the second example, no correspondence leads to a correct list: in the list given in the input, there must be three Mondays during ten consecutive days regardless of the correspondence.

Problem E. Paths in tree

Input file:	paths-in-tree.in
Output file:	paths-in-tree.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	**512** mebibytes

Consider a tree with n vertices. You have to select at most k simple paths without intersections by vertices. The total length of these paths should be the maximal possible. In this problem, the length of a path is the number of vertices in it.

Input

Each input contains one or more test cases.

Each test case starts with a line containing two integers n and k ($1 \le n \le 10\,000$, $1 \le k \le 500$, $k \le n$). It is followed by $n - 1$ lines which contain pairs of integers from 1 to n: edges of the tree. The sum of all n does not exceed 10 000.

The input ends with pair $n = 0$, $k = 0$. This pair should not be considered a test case.

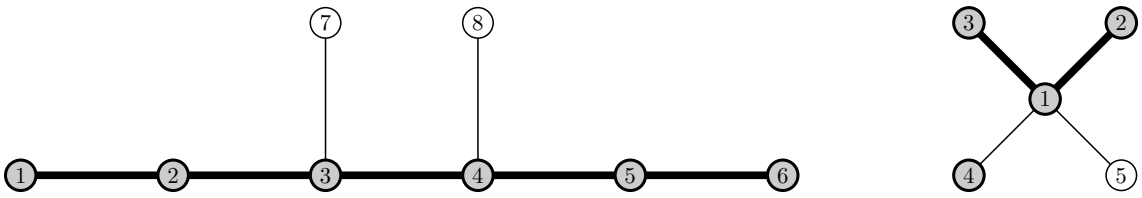
Output

For each test case, print the total length of selected paths on a separate line (it is an integer from 0 to n). On the next line, print integer x ($0 \le x \le k$): the number of paths in the optimal answer you found. Next x lines should contain pairs of integers from 1 to n: endpoints of paths in the optimal answer you found. Endpoints of a single path may coincide: in this case, the path consists of exactly one vertex. Note that each simple path in a tree is uniquely determined by its endpoints. If there are several optimal answers, you may output any one of them.

Example

paths-in-tree.in	paths-in-tree.out
8 1 1 2 2 3 3 4 4 5 5 6 3 7 4 8 5 2 1 2 1 3 1 4 1 5 0 0	6 1 6 1 4 2 2 3 4 4

Explanation



Problem F. Quartet Distance

Input file:	quartets.in
Output file:	quartets.out
Time limit:	3 seconds (*4 seconds for Java*)
Memory limit:	256 mebibytes

An *unrooted binary tree* is a tree with labeled leaves and unlabeled internal nodes with degree of all internal nodes equal to three. It can be proved that an unrooted binary tree with n leaves has exactly $2n - 3$ edges.

Such trees are widely used in biology to display evolutionary relationships between species. Leaves represent living species which we can observe and internal nodes represent extinct ancestors of which we have no information.

One of the most common tasks is to find how similar are the two trees over the same set of species.

Consider four leaves, A, B, C, and D. They form a quartet $AB|CD$ in a tree T if there exists an edge that separates T into two subtrees T_1 and T_2 such that $A \in T_1$, $B \in T_1$, $C \in T_2$ and $D \in T_2$. Quartets are compared as partitions of a set of four vertices into two sets of two vertices in each. For example, quartets $AB|CD$, $BA|CD$, $DC|AB$ are all equal, whereas quartets $AB|CD$ and $AC|BD$ are different. Only leaves can form a quartet.

Let us denote the set of all quartets of a tree as $Q(T)$. Then we define the *quartet distance* between T_1 and T_2 as $d_q(T_1, T_2) = |Q(T_1) \triangle Q(T_2)|$. Here, $A \triangle B$ is symmetric difference between sets A and B, that is, the set of elements which are either in A or in B, but not in both.

Given two trees, find the quartet distance between them.

Input

The first line of input contains an integer n, the number of leaves in each tree ($4 \le n \le 1000$). Then come $2n - 3$ edges of the first tree, and after that, $2n - 3$ edges of the second tree. Each edge occupies one line and contains a pair of numbers of vertices that it connects. Leaves are numbered by integers from 1 to n. Internal nodes are numbered by integers from $n + 1$ to $2n - 3$.

Output

On the first line, print the quartet distance $d_q(T_1, T_2)$.

Example

quartets.in	quartets.out
5 1 7 2 6 3 6 4 8 5 8 6 7 7 8 1 6 2 7 3 6 4 8 5 8 6 7 7 8	4

Explanation

In the example, $Q(T_1) = \{23|14, 23|15, 23|45, 45|12, 45|13\}$, $Q(T_2) = \{13|24, 13|25, 13|45, 45|12, 45|23\}$, $Q(T_1) \triangle Q(T_2) = \{23|14, 23|15, 13|24, 13|25\}$ and the distance is equal to 4.

Problem G. Sheffer Stroke

Input file:	sheffer.in
Output file:	sheffer.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Vasya has a calculator for boolean values: numbers 0 and 1. This calculator can perform only one operation: Sheffer stroke ("|"). The truth table for this operation showing its result for every possible pair of arguments is given below.

x	y	x|y
0	0	1
0	1	1
1	0	1
1	1	0

Vasya has a boolean function F of n arguments: he knows all 2^n values this function should take for every possible collection of arguments. Vasya is given a task to write it in reverse Polish notation for his calculator. Help him do it.

The reverse Polish notation for this problem can be defined recursively as follows:

- The arguments of the function are denoted by first n lowercase English letters: "a", "b", ... Each of these letters is a correct expression in reverse Polish notation, and its value is the value of the argument with the corresponding number.
- If A and B are correct expressions in reverse Polish notation, then "AB|" is also a correct expression in reverse Polish notation, and its value can be computed as Sheffer stroke applied to the values of expressions A and B.
- There are no other correct expressions.

Input

The first line of input contains 2^n binary digits without any spaces. This line gives the values of the function for every possible collection of arguments. The values are listed in the order for which the collections of arguments are ordered lexicographically. For example, if $n = 2$, the values are listed in the following order: $F(0,0)$, $F(0,1)$, $F(1,0)$, $F(1,1)$.

It is guaranteed that $1 \le n \le 6$.

Output

On the first line, print a string consisting of one to 500 000 characters: the reverse Polish notation of function F for Vasya's calculator. This string can contain only the first n lowercase English letters and symbol "|" (ASCII-code 124). If there are several possible answers, you can print any one of them. Note that it is not necessary to minimize the length of your answer.

It is guaranteed that, for each function F which is possible according to the statement, there exists an answer within the given constraints.

Example

sheffer.in	sheffer.out
1110	ab|

Explanation

In this example, the given function F of two arguments is exactly the Sheffer stroke: $F(0,0) = 1$, $F(0,1) = 1$, $F(1,0) = 1$ and $F(1,1) = 0$. Thus the following holds: $F(\mathtt{a},\mathtt{b}) = \mathtt{a|b}$. The expression "a|b" written in reverse Polish notation looks like this: "ab|".

Problem H. Snakes

Input file:	`snakes.in`
Output file:	`snakes.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Snakes of Mistress of the Copper Mountain crawled out to bask in the sun. They moved onto a malachite stone which appeared to be a rectangle of size $m \times n$. For aesthetic reasons, they occupied positions according to the pattern on the malachite so that the tail tip of each snake touches its head.

The pattern on the malachite introduces a rectangular grid dividing the stone into $m \times n$ square cells. Each snake lying on the stone forms a closed polygonal curve. The vertices of that curve are located at centers of cells, and each two consecutive vertices are located at two cells which share a side. Thus each snake divides the malachite into two parts: the part *inside* the snake and the exterior part. The snakes chose their positions in such a way that, for every cell of the stone, there is exactly one snake which passes through that cell.

Snake A *surrounds* snake B if the closed polygonal curve formed by A contains the curve formed by B inside, and in the case there are other snake curves contained inside the curve formed by A, they are all contained inside B. Note that each snake can surround at most one other snake, and each snake can be surrounded by at most one other snake.

If snake A_1 surrounds snake A_2, A_2 surround A_3, ..., A_{s-1} surrounds A_s and additionally, no snake surrounds A_1 and no snake is surrounded by A_s, such sequence of snakes is called a *chain*. Note that each snake is a part of exactly one chain. For example, a snake which neither surrounds nor is surrounded by any other snake constitutes a chain consisting of that snake.

When the sun hides behind a cloud, snakes will go home in chains. Mistress of the Copper Mountain is interested in the chains formed by her snakes. She also wants to know the order of snakes in each chain, from the outermost to the innermost. Help her find this information.

Input

The first line of input contains three integers m, n and k: the dimensions of the malachite and the number of snakes ($2 \le m, n \le 100$, $1 \le k \le \frac{m \cdot n}{4}$). Each of the following k lines describes one snake. The description of snake i starts with an integer l_i: the length of the snake ($4 \le l_i \le m \cdot n$). It is followed by l_i pairs of integers $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$, ..., x_{i,l_i}, y_{i,l_i}: the coordinates of cells which are passed by the snake, given in the order of traversal ($1 \le x_{i,j} \le m$, $1 \le y_{i,j} \le n$).

It is guaranteed that for every cell of the stone, there is exactly one snake which passes through that cell, and that the sequence of coordinates of each snake constitutes a closed polygonal curve in which each two consecutive vertices share a side.

Output

On the first line of output, print an integer q: the number of chains. On the each of following q lines, print one of the chains. The description of chain i starts with an integer s_i: the number of snakes in i-th chain. Then must follow s_i integers: the numbers of snakes in the chain from the outermost to the innermost. Snakes are numbered from one in the order in which they are given in the input.

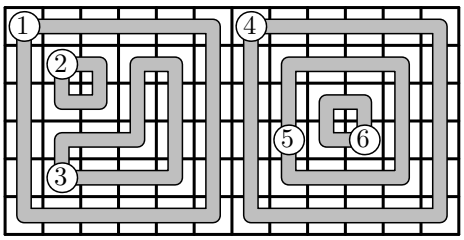
You can print chains in any order.

Example

snakes.in
12 6 6 20 1 1 1 2 1 3 1 4 1 5 1 6 2 6 3 6 4 6 5 6 6 6 6 5 6 4 6 3 6 2 6 1 5 1 4 1 3 1 2 1 4 2 2 2 3 3 3 3 2 12 2 5 2 4 3 4 4 4 4 3 4 2 5 2 5 3 5 4 5 5 4 5 3 5 20 7 1 7 2 7 3 7 4 7 5 7 6 8 6 9 6 10 6 11 6 12 6 12 5 12 4 12 3 12 2 12 1 11 1 10 1 9 1 8 1 12 8 4 8 5 9 5 10 5 11 5 11 4 11 3 11 2 10 2 9 2 8 2 8 3 4 10 4 10 3 9 3 9 4
snakes.out
4 1 1 1 2 1 3 3 4 5 6

Explanation

In this example, the first snake contains the second and the third snakes inside, but does not surround any of them because there are two of them and no one contains the other. The second and the third snakes also do not surround anyone. The fourth snake contains the fifth and the sixth ones inside. As the fifth snake contains the sixth one, we see that the fourth snake *surrounds* the fifth one. The latter, in turn, *surrounds* the sixth one, and the sixth snake does not surround anyone.



The first three snakes constitute three separate chains. The fourth, fifth and sixth snakes constitute another chain.

Problem I. Breaking Hashing (Division 2)

Input file:	`breaking-hashing.in`
Output file:	`breaking-hashing.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Your solutions fail on corner cases?
Built-in quick sort suddenly works in quadratic time?
You struggle hard against precision losses in geometric problems?
Your solutions pass local stress testing but fail on the judges' tests?
The bug turned out to be in the library routine and not in your solution?

You want to know who is behind all of that?

Today you have a unique opportunity to join a secret organization:
the Order of Cunning Beavers! Members of this organization make approximately 146%
more successful hacks than uninitiated folks, and in the problems authored by them,
the average number of missed test cases is significantly less. To apply for
junior membership, one has to pass the examination: solve the problem below.

Hurry! Only the first few will be able to apply!

In this problem, it is required to find a collision for polynomial hashing of strings consisting of lowercase English letters.

A *polynomial hash* of a string has two parameters: multiplier p and modulus q. For empty string ε, the value of hash function $h(\varepsilon) = 0$, and for any string S and character c it is defined recursively by the following formula: $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Here, $\text{code}(c)$ is the ASCII code of character c. Recall that codes of lowercase English letters are consecutive: code('a') = 97, code('b') = 98, ..., code('z') = 122. We can also obtain a general formula: if $S = s_1 s_2 \dots s_n$, we have $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

Given the parameters p and q, find two different non-empty strings A and B such that $h(A) = h(B)$.

Input

The first line of input contains two integers p and q separated by a space: the parameters of the hash function ($0 < p < q < 2 \cdot 10^9$).

Output

On the first two lines, output two different non-empty strings A and B such that $h(A) = h(B)$. The strings should consist entirely of lowercase English letters (ASCII codes 97–122) and have lengths from 1 to 100 000 characters. Note that lenghts of the strings are not required to be equal. In case of multiple answers, output any one of them.

Example

breaking-hashing.in	breaking-hashing.out
31 47	aa bq

Explanation

In the example, $h(A) = (97 \cdot 31 + 97) \bmod 47 = 3104 \bmod 47 = 2$ and
$h(B) = (98 \cdot 31 + 113) \bmod 47 = 3151 \bmod 47 = 2$.

Problem J. Paths in Tree (Division 2)

Input file:	`paths-in-tree.in`
Output file:	`paths-in-tree.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	**512** mebibytes

Consider a tree with n vertices. You have to select at most k simple paths without intersections by vertices. The total length of these paths should be the maximal possible. In this problem, the length of a path is the number of vertices in it.

Input

Each input contains one or more test cases.

Each test case starts with a line containing two integers n and k ($1 \le n \le 10\,000$, $1 \le k \le 100$, $k \le n$). It is followed by $n - 1$ lines which contain pairs of integers from 1 to n: edges of the tree. The sum of all n does not exceed 10 000.

The input ends with pair $n = 0$, $k = 0$. This pair should not be considered a test case.

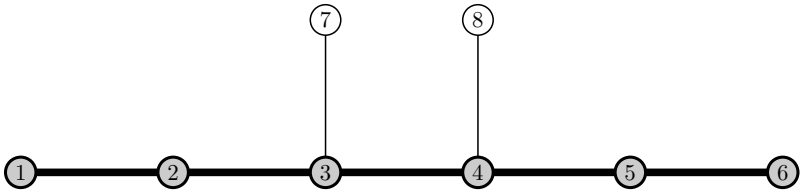
Output

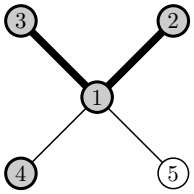
For each test case, print the total length of selected paths on a separate line (it is an integer from 0 to n).

Example

paths-in-tree.in	paths-in-tree.out
8 1 1 2 2 3 3 4 4 5 5 6 3 7 4 8 5 2 1 2 1 3 1 4 1 5 0 0	6 4

Explanation





Problem K. Quartet Distance (Division 2)

Input file:	quartets.in
Output file:	quartets.out
Time limit:	1 second (*1.5 seconds for Java*)
Memory limit:	256 mebibytes

An *unrooted binary tree* is a tree with labeled leaves and unlabeled internal nodes with degree of all internal nodes equal to three. It can be proved that an unrooted binary tree with n leaves has exactly $2n - 3$ edges.

Such trees are widely used in biology to display evolutionary relationships between species. Leaves represent living species which we can observe and internal nodes represent extinct ancestors of which we have no information.

One of the most common tasks is to find how similar are the two trees over the same set of species.

Consider four leaves, A, B, C, and D. They form a quartet $AB|CD$ in a tree T if there exists an edge that separates T into two subtrees T_1 and T_2 such that $A \in T_1$, $B \in T_1$, $C \in T_2$ and $D \in T_2$. Quartets are compared as partitions of a set of four vertices into two sets of two vertices in each. For example, quartets $AB|CD$, $BA|CD$, $DC|AB$ are all equal, whereas quartets $AB|CD$ and $AC|BD$ are different. Only leaves can form a quartet.

Let us denote the set of all quartets of a tree as $Q(T)$. Then we define the *quartet distance* between T_1 and T_2 as $d_{\mathrm{q}}(T_1, T_2) = |Q(T_1) \triangle Q(T_2)|$. Here, $A \triangle B$ is symmetric difference between sets A and B, that is, the set of elements which are either in A or in B, but not in both.

Given two trees, find the quartet distance between them.

Input

The first line of input contains an integer n, the number of leaves in each tree ($4 \le n \le 70$). Then come $2n - 3$ edges of the first tree, and after that, $2n - 3$ edges of the second tree. Each edge occupies one line and contains a pair of numbers of vertices that it connects. Leaves are numbered by integers from 1 to n. Internal nodes are numbered by integers from $n + 1$ to $2n - 3$.

Output

On the first line, print the quartet distance $d_{\mathrm{q}}(T_1, T_2)$.

Example

quartets.in	quartets.out
5 1 7 2 6 3 6 4 8 5 8 6 7 7 8 1 6 2 7 3 6 4 8 5 8 6 7 7 8	4

Explanation

In the example, $Q(T_1) = \{23|14,\ 23|15,\ 23|45,\ 45|12,\ 45|13\}$, $Q(T_2) = \{13|24,\ 13|25,\ 13|45,\ 45|12,\ 45|23\}$, $Q(T_1) \triangle Q(T_2) = \{23|14,\ 23|15,\ 13|24,\ 13|25\}$ and the distance is equal to 4.

Problem L. Sheffer Stroke (Division 2)

Input file:	`sheffer.in`
Output file:	`sheffer.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Vasya has a calculator for boolean values: numbers 0 and 1. This calculator can perform only one operation: Sheffer stroke ("`|`"). The truth table for this operation showing its result for every possible pair of arguments is given below.

x	y	x|y
0	0	1
0	1	1
1	0	1
1	1	0

Vasya has a boolean function F of n arguments: he knows all 2^n values this function should take for every possible collection of arguments. Vasya is given a task to write it in reverse Polish notation for his calculator. Help him do it.

The reverse Polish notation for this problem can be defined recursively as follows:

- The arguments of the function are denoted by first n lowercase English letters: "`a`", "`b`", ... Each of these letters is a correct expression in reverse Polish notation, and its value is the value of the argument with the corresponding number.
- If `A` and `B` are correct expressions in reverse Polish notation, then "`AB|`" is also a correct expression in reverse Polish notation, and its value can be computed as Sheffer stroke applied to the values of expressions `A` and `B`.
- There are no other correct expressions.

Input

The first line of input contains 2^n binary digits without any spaces. This line gives the values of the function for every possible collection of arguments. The values are listed in the order for which the collections of arguments are ordered lexicographically. For example, if $n = 2$, the values are listed in the following order: $F(0,0)$, $F(0,1)$, $F(1,0)$, $F(1,1)$.

It is guaranteed that $1 \le n \le 4$.

Output

On the first line, print a string consisting of one to 500 000 characters: the reverse Polish notation of function F for Vasya's calculator. This string can contain only the first n lowercase English letters and symbol "`|`" (ASCII-code 124). If there are several possible answers, you can print any one of them. Note that it is not necessary to minimize the length of your answer.

It is guaranteed that, for each function F which is possible according to the statement, there exists an answer within the given constraints.

Example

sheffer.in	sheffer.out
1110	ab|

Explanation

In this example, the given function F of two arguments is exactly the Sheffer stroke: $F(0,0) = 1$, $F(0,1) = 1$, $F(1,0) = 1$ and $F(1,1) = 0$. Thus the following holds: $F(\mathtt{a},\mathtt{b}) = \mathtt{a|b}$. The expression "`a|b`" written in reverse Polish notation looks like this: "`ab|`".