Problem A. Almost Palindromic Numbers

Input file:	`almost.in`
Output file:	`almost.out`
Time limit:	1 second
Memory limit:	64 megabytes

Let us call an integer number x *almost palindromic* if its decimal representation can be converted to a palindrome by changing at most one digit. For example, 1234321, 1234311, 123421 are almost palindromic, but 1234213 or 12345331 are not.

Given a, find the number of almost palindromic numbers x such that $1 \le x \le a$.

Input

Input file contains several test cases. Each test case consists of a single integer number a on a line by itself ($1 \le a \le 10^{18}$). The last test case is followed by a line containing single zero, this line must not be processed.

Output

For each number a in the input file print the number of almost palindromic numbers not exceeding a.

Example

almost.in	almost.out
10 1000 1000000 0	10 1000 45010

Problem B. Cartesian Tree

Input file:	`cartesian.in`
Output file:	`cartesian.out`
Time limit:	1 second
Memory limit:	64 megabytes

Let us consider a special type of binary search trees, called *cartesian trees*. Recall that a binary search tree is a rooted ordered binary tree, such that for its every node u the following condition is satisfied: each node in its left subtree has the key less than the key of u, and each node in its right subtree has the key greater than the key of u.

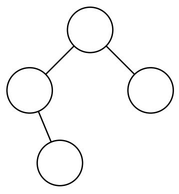
That is, if we denote the left subtree of the node u by $L(u)$, its right subtree by $R(u)$ and its key by x_u, for each node u we will have: if $v \in L(u)$ then $x_v < x_u$, if $w \in R(u)$ then $x_w > x_u$.

The binary search tree is called *cartesian* if its every node u in addition to the main key x_u also has an auxiliary key y_u, and for these keys the *heap condition* is satisfied, that is: if v is the parent of u then $y_v < y_u$.

Let us call a set of pairs $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$ *correct* if all x_i are distinct numbers from 1 to n, and all y_i are distinct numbers from 1 to n as well. It can be easily proved that if we have a correct set of pairs there is a unique cartesian tree that can be constructed of them.

Consider a rooted ordered binary tree T with n nodes. Your task is to find the number of correct sets of pairs, such that there is a way to put them to nodes of T to create a cartesian tree.

For example, there are three sets of pairs that can be put to nodes of the tree, shown on the picture below: $\{(1,2),(2,3),(3,1),(4,4)\}$, $\{(1,2),(2,4),(3,1),(4,3)\}$ and $\{(1,3),(2,4),(3,1),(4,2)\}$.



Input

The first line of the input file contains n — the number of nodes in a tree T ($1 \le n \le 200$). The next n lines describe its nodes. Each node is described with two numbers — the number of its left child, and the number of its right child. If there is no respective child, the corresponding number is 0. It is guaranteed that the description of the tree is correct. The root of the tree has number 1.

Output

Output one number — the number of correct sets of pairs that have T as their cartesian tree.

Example

`cartesian.in`	`cartesian.out`

4
2 3
0 4
0 0
0 0
	3

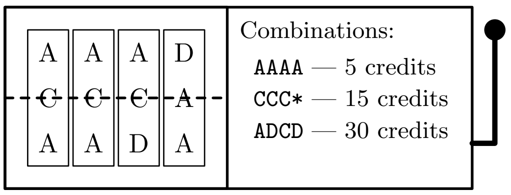
Problem C. Casino

Input file:	`casino.in`
Output file:	`casino.out`
Time limit:	1 second
Memory limit:	64 megabytes

Some casinos in Las Vegas probably make more money by slot machines than by other means of gambling. Let us describe a classical Las Vegas slot machine.

The slot machine has n windows, each shows k characters from the m-character wheel. Each window has its own wheel. Usually various funny pictures are used as characters, but in this problem we will use letters of the English alphabet.

The gambler pays one credit, pulls the handle, and the wheels start to rotate randomly. After they stop, the machine shows how much the gambler has won. The winnings are determined using the following procedure. After the wheel stops some of its characters are visible through the window. A simple slot machine has one line going across all windows, identifying exactly one character at each wheel. If these characters form a combination, the gambler wins some money.

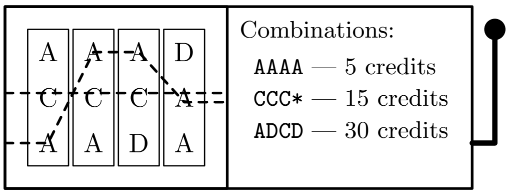


For example, let there be four wheels, each containing 6 characters: '`A`', '`A`', '`C`', '`A`', '`C`' and '`D`', in this order. Let each window show three characters, and the line goes through central characters in all windows. Let the combinations be "`AAAA`" for 5 credits, "`CCC*`" for 15 credits, and "`ADCD`" for 30 credits. Here '`*`' denotes any character.

Let the machine show "`ACA`" in the first window, "`ACA`" in the second window, "`ACD`" in the third window, and "`DAA`" in the fourth one. In this case the characters on the line form "`CCCA`" and the player wins 15 credits.

However, the game with just one winning line is not interesting. Winnings are rare and the player soon gets bored. Therefore the concept of *multiple lines* is introduced in more complicated slot machines. A *line* is the sequence $\langle i_1, i_2, \ldots, i_n \rangle$ of indices from 1 to k that tells which character from each window should be taken to check for combination on this line. There can be several lines in the slot machine, the gambler may choose several of them before pulling the handle, and she would get a winning for a combination on each line. Of course, she must pay one credit for each line she plays on.

For example, if in the machine described above in addition to a standard line $\langle 2, 2, 2, 2 \rangle$ there were a line $\langle 3, 1, 1, 2 \rangle$, the gambler played on both lines and got the above outcome, she would also get a combination "`AAAA`" on the second line and win additional 5 credits.



You are planning to go to Las Vegas and need to prepare a good strategy for playing on a multi-line slot machines. Given a description of a slot machine, you have to determine which lines should one choose to play on to maximize winning ratio. Winning ratio is calculated as the sum of expected winnings on selected lines divided by the number of lines.

Input

The first line of the input file contains five integer numbers: n, m, k, c and l — the number of wheels, characters on a wheel, characters in a window, winning combinations, and lines ($1 \le n, m \le 40$, $1 \le k \le m$, $1 \le c \le 20$, $1 \le l \le 20$).

The following n lines describe wheels. Each wheel is described by m characters in order they appear on a wheel. You may assume that the machine is honest, that is — each block of k consecutive characters has equal probability of appearance in a window.

The following c lines describe combinations. Each combination is described by n characters needed for this combination and the winning in credits (‘`*`’ denotes any character for the corresponding wheel). The winning is a positive integer number not exceeding 10^6.

The following l lines contain n integer numbers each and describe lines.

Output

Print z — the optimal winning ratio — at the first line of the output file. Print it as an irreducible fraction in a form “`numerator/denominator`”. The second line must contain p — the number of lines that must be used to achieve this ratio. The third line must contain p numbers — the lines to use.

It is possible that the expected winning for any set of lines is not positive, in this case it is better not to play at all, z is `0/1`, and p is 0.

Example

casino.in	casino.out
4 6 3 3 2 AACACD AACACD AACACD AACACD AAAA 5 CCC* 15 ADCD 30 2 2 2 2 3 1 1 2	1/144 2 1 2

Problem D. DNA Analysis

Input file:	`dna.in`
Output file:	`dna.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Biologists of Karelia Mutation Project (KMP) have recently started a new project that is intended to prove that humans and mammoths are close relatives. To prove this statement, the biologists are planning to carry out the following experiment.

The human DNA is compared to the mammoth DNA. The DNA is split into pieces of length n, and pieces are consecutively compared. Since some mutations are possible, the comparison is carried out as following.

Consider a string α. We say that α *mutates* to β if $\alpha = xyz$ for some (possibly empty) strings x, y and z, and $\beta = xy^Rz$ where y^R is the reversed string y (for example, “`abc`”^{R} =“`cba`”). We say that string α and β are *similar* if α can be transformed to β with at most 4 mutations.

Given two pieces of DNA, check whether they are similar.

Input

The input file contains two strings, containing only characters ‘`A`’, ‘`D`’, ‘`G`’ and ‘`T`’. The strings have the same length, it doesn’t exceed 30.

Output

Output “`Similar`” if the strings are similar, and “`Different`” if they are not.

Example

dna.in	dna.out
`ATGAATGA` `AGGAATTA`	`Similar`
`ATGAATGAATGA` `TTTAAAAAAGGG`	`Different`

In the first example mutations can proceed as follows: “`ATGAATGA`”$\rightarrow$“`AGTAAGTA`”$\rightarrow$“`AGGAATTA`”.

Problem E. Formula 1

Input file:	`formula1.in`
Output file:	`formula1.out`
Time limit:	1 second
Memory limit:	64 megabytes

Michael has watched Formula 1 on TV today. He was so impressed by the fast cars, that immediately decided to create his own Formula 1 team. So he and his brother Freddie started to design the car.

They know that the car has many various settings. For their simplified initial model of the car they have developed n settings. Each setting can take a real value from 0 to 1, inclusive. The total power of the car is expressed as the polynomial of the settings with real coefficients. Since brothers has just finished the seventh grade, the degree of each monomial in a polynomial is at most 2.

Now they want to choose such settings that the total power of the car is maximal possible. Help them to do it.

Input

The input file contains the expression of the total power of the car as a polynomial of its settings. The degree of the polynomial is at most two. The polynomial is specified as the sum of monomials, each monomial is specified as a real coefficient followed by zero, one or two variables. Variables represent car settings, each variable is a small letter of the English alphabet. As usually, negative coefficients are not preceded by '+', coefficients equal to 1 are omitted, coefficients equal to -1 are specified as just '-'. Square of a variable `x` is denoted as "`x^2`".

The car that brothers are designing has at most 8 settings. No coefficients exceed 10^4 by their absolute value. The length of the polynomial doesn't exceed 10000 characters.

Output

The first line of the output file must contain the maximal possible total power of the car. After that print the optimal value for each setting. If there are several solutions, output any one. Adhere to the format of sample output. You answer must be accurate up to 10^{-5}.

Example

formula1.in	formula1.out

`-2x^2+x+0.8x-2y^2+y+0.1xy`	`0.541588`
	`x=0.456535`
	`y=0.261413`

`-4`	`-4.0`

`x-x`	`0.0`
	`x=0.0`

Problem F. Surface Genus

Input file:	`genus.in`
Output file:	`genus.out`
Time limit:	1 second
Memory limit:	64 megabytes

Consider a compact connected orientable two-dimensional surface without boundary in a three-dimensional space. Examples of such surfaces include sphere or torus. A nice fact comes from topology which says that every such surface is homeomorphic to a sphere with several “handles”. Two surfaces are called homeomorphic if one can be transformed to another by continuous deformations.

The number of handles that the corresponding homeomorphic sphere has is called a *genus* of the surface. For example, sphere is clearly homeomorphic to a sphere with zero handles, therefore its genus is zero. A torus is homeomorphic to a sphere with one handle, therefore its genus is one.

You are given a triangulation of some surface in three-dimensional space. Your task is to find its genus.

Input

The first line of the input file contains n — the number of triangles in the triangulation ($1 \le n \le 5000$). The rest of the input file describes triangles. Each triangle is specified with 9 integer numbers: three triples of the coordinates of its vertices.

It is guaranteed that the given triangles form the triangulation of a compact connected surface without boundary, no two triangles intersect other than by their boundaries. No vertex of a triangle is an internal point of an edge of another triangle. All coordinates do not exceed 10^6 by their absolute values.

Output

Output one number — the genus of the surface specified in the input file.

Example

genus.in	genus.out

```
4
0 0 0  10 0 0  5 7 0
0 0 0  10 0 0  5 3 7
10 0 0  5 7 0  5 3 7
0 0 0  5 7 0  5 3 7
```
	0

```
24
0 0 0  3 2 0  4 7 0   0 0 0  3 2 0  5 2 0
0 0 0  5 2 0  8 0 0   3 2 0  4 4 0  4 7 0
4 4 0  4 7 0  8 0 0   5 2 0  4 4 0  8 0 0
0 0 1  3 2 1  4 7 1   0 0 1  3 2 1  5 2 1
0 0 1  5 2 1  8 0 1   3 2 1  4 4 1  4 7 1
4 4 1  4 7 1  8 0 1   5 2 1  4 4 1  8 0 1
0 0 0  0 0 1  4 7 0   0 0 1  4 7 1  4 7 0
8 0 0  8 0 1  4 7 0   8 0 1  4 7 1  4 7 0
8 0 0  8 0 1  0 0 0   8 0 1  0 0 1  0 0 0
3 2 0  3 2 1  4 4 0   3 2 1  4 4 1  4 4 0
5 2 0  5 2 1  4 4 0   5 2 1  4 4 1  4 4 0
5 2 0  5 2 1  3 2 0   5 2 1  3 2 1  3 2 0
```
	1

Problem G. Matrix Multiplication

Input file:	`matrix.in`
Output file:	`matrix.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Little Josh learns how to multiply matrices. He practices by multiplying large binary matrices over $\mathbb{F}_2$ (the operations in this field are performed modulo 2). Recently he has multiplied two $n \times n$ matrices A and B, and now claims that the result is C.

But Jenny doesn't believe him. She says that he is wrong. Help the children to resolve their dispute. Given matrices A, B and C, check whether indeed $AB = C$.

Input

The first line of the input file contains n — the size of matrices ($1 \le n \le 4000$). The following 3 lines describe matrices A, B and C.

Each matrix is described by a line that contains n blocks of $\lceil n/4 \rceil$ hexadecimal digits. If you write the digits in a binary notation, higher bits first, and truncate extra digits in the end, you will get the corresponding row of the matrix. For example, the matrix

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

is described as `28`, `A8`, `68`, `78`, `D0`, `88`.

Output

Output "`YES`" if $AB = C$ is true, and "`NO`" otherwise.

Example

`matrix.in`	`matrix.out`

`6`
`28 A8 68 78 D0 88`
`80 40 20 10 08 04`
`28 A8 68 78 D0 88`
	`YES`

`6`
`28 A8 68 78 D0 88`
`80 40 20 10 08 04`
`28 B8 68 78 D0 88`
	`NO`

Note

Input files in this problem are huge. Please do not submit a solution if you understand that it will not pass because of the time limit — do not create testing jams.

Yes, judges have a solution for this problem in Java that runs well under time limit. Do not use "`std::ifstream`" or "`java.util.Scanner`" to read the input file in this problem.

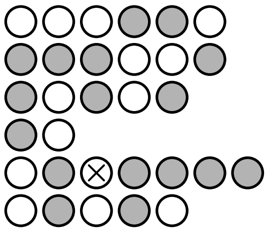
Problem H. Colorful Pebbles

Input file:	`pebbles.in`
Output file:	`pebbles.out`
Time limit:	1 second
Memory limit:	64 megabytes

Andrew and Mike play the following game with pebbles. They put several rows of blue and red pebbles to the table. Andrew owns the blue pebbles, Mike owns the red ones.

The players make moves in turn. Andrew moves first. Each turn the player chooses any pebble of his color and removes it together with all pebbles located to the right of the removed pebble in its row. Although the chosen pebble must be of player's color, the pebbles to the right are removed from the table regardless of their color. The player who has no pebbles of his color left before his move loses.

Let us consider an example shown on the picture below. Blue pebbles are shown as empty circles, red pebbles are shown as shaded ones.



Here Andrew can win. In this first turn he must take the third pebble in the fifth row (the one marked with a small cross). After that he must use a symmetric strategy: if Mike moves in the first row, Andrew moves the same way in the second one, if Mike moves in the second row — Andrew repeats his move in the first, Mike moves in the third row — Andrew does in the sixth, and so on.

Given the initial position of pebbles on the table, find out who wins if both play optimally, and if it is Andrew find his winning first move.

Input

The first line of the input file contains n — the number of rows of pebbles ($1 \le n \le 100$). The following n lines describe rows. Each of these lines contains a sequence of '`B`' and '`R`' characters standing for blue and red pebbles, respectively. Each row contains at most 20 pebbles.

Output

Print "`Andrew`" or "`Mike`" at the first line of the output file, depending on the one who wins the game. If Andrew can win, print the number of the row, and the position of the pebble in that row, that Andrew must take in his first move to win.

Example

`pebbles.in`	`pebbles.out`

`6`
`BBBRRB`
`RRRBBR`
`RBRBR`
`RB`
`BRBRRRR`
`BRBRB`
	`Andrew`
	`5 3`

Problem I. Princess Dilemma

Input file:	`princess.in`
Output file:	`princess.out`
Time limit:	1 second
Memory limit:	64 megabytes

Once upon a time in the kingdom far away there lived a princess. And she was so beautiful that every young man in the kingdom wanted to marry her. But the princess was very haughty and didn't want to marry the first man she meets. Therefore she decided that she would marry only the most beautiful young man in the kingdom.

But there was a little problem — how to select the most beautiful one? Of course, had all young people gathered together in front of the princess, she would immediately tell who is the most beautiful. But the young men are usually rather busy playing, fighting, and doing other very important things. Therefore all she could was to ask them to come one person a day, and choose the one to marry.

Fortunately, the princess had a very good memory. After seeing a man she could always tell for any other man, whether he was more beautiful, or less beautiful. The princess beauty criteria were so exact, that no two young men in the kingdom were equally beautiful for her.

The princess decided to do the following. She created the list of all n beautiful young men and ordered it randomly. After that she would invite young men to her castle one after another in the order of the list. After meeting each man she would decide whether she would marry him, or not. After she married one man, she would continue to invite other young men to make sure she married the most beautiful one. If it would indeed be so, they would live happily until the end of the days. In the other case, if she married the man who was not the most beautiful, or didn't marry at all, the princess would commit suicide.

Help the princess to develop the best strategy, so that she marries the most beautiful young man with the maximal possible probability. The only thing that the princess can do to make a decision whether to marry the current young man is to compare his beauty to the beauty of all previous young men.

Input

The input file contains one integer number n ($1 \le n \le 2006$).

Output

Output one real number — the probability that the princess would marry the most beautiful young man if she follows the optimal strategy. The probability is taken over all possible young men orderings. The princess strategy must be deterministic.

Your answer must be accurate up to 10^{-8}.

Example

princess.in	princess.out
1	1.0
2	0.5
3	0.5

In the last example the princess must act as follows. The first young man must be rejected. The second young man must be accepted if and only if he is more beautiful than the first one. Using this strategy the princess fails to marry the most beautiful young man for the following permutations of beauty (the greater — the more beautiful): $\langle 1, 2, 3\rangle$, $\langle 3, 1, 2\rangle$, $\langle 3, 2, 1\rangle$, and succeeds for all other permutations. All other strategies give a smaller success probability.

Problem J. Crossing the River

Input file:	`river.in`
Output file:	`river.out`
Time limit:	1 second
Memory limit:	64 megabytes

Josh and Jill went for a walk. They started their way from Josh's home at the West end of the long road, and finished it near Jill's home at the road's East end. While walking, they noticed, that they have crossed the river Curvy n times.

Unfortunately, none of the kids remembers how the river Curvy flows. Josh says that it flows from South to North, but Jill says that it flows from North to South. Their argue is ready to turn to a fight when Jill's father comes home. After listening to children's arguments, he recommends them to count how many different river configurations are possible.

But the problem seems too difficult for them. You must help.

Input

Input file contains one integer number n ($1 \le n \le 16$).

Output

Output one integer number — the number of river configurations possible.

Example

`river.in`	`river.out`
`1`	`2`
`2`	`4`
`3`	`4`
`4`	`12`

All possible configurations for the last example are shown on the picture below. Please note, that the variants that differ by the river direction are considered different.