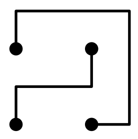
Problem A. Chip Designing

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

Designing computer chips is quite difficult. Recently Peter has decided to get a job in *Karelia Quaterconductors* company that is going to produce chips for new *EBM* processor. As a qualifying work for the position he has got a task of designing a layout for a very simple chip. A chip has n inputs and n outputs, located in points $(0, 0)$, $(1, 0)$, $\ldots$, $(n-1, 0)$ and $(0, 1)$, $(1, 1)$, $\ldots$, $(n-1, 1)$ respectively (all dimensions are, of course, in nanometers).

In this chip the i-th input must be connected to the a_i-th output using nanowires. All nanowires must be polylines with segments parallel to the sides of the chip (coordinate axes). No two nanowires must intersect (there must be no common point for any two nanowires). The thickness of the wires is so small, that it can be ignored.

Help Peter to get the job, write a program that will design the chip for him.



Input

The first line of the input file contains n ($1 \le n \le 10$). Next line contains n different integer numbers — $a_1, a_2, \ldots, a_n$.

Output

Output the description of n nanowires. Each description must start with k — the number of segments in the wire (k must not exceed 1000). Next $k + 1$ lines must contain two numbers each and specify the coordinates of the breaking points of the polyline. All coordinates must be specified as rational irreducible fractions in a form "*nominator*/*denominator*". Nominator and denominator must not exceed 10^9.

The first point of the i-th polyline must be the i-th input of the chip and its last point must be the a_i-th output of the chip.

Example

chip.in	chip.out
2 2 1	3 0/1 0/1 0/1 1/2 1/1 1/2 1/1 1/1 4 1/1 0/1 3/2 0/1 3/2 3/2 0/1 3/2 0/1 1/1

Problem B. Electricity

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

After recent blackouts in some regions in North America, the government has decided to reorganize the power supply network of the continent.

The power supply network is the set of nodes, such as power plants or transformation stations, connected by transmission lines. All lines are used to transmit electricity from one node to another. For stability reasons the system is organized in such a way that there are no directed cycles.

Since the government is currently short of money due to several small peaceful militaristic operations, it cannot build new power lines for the moment. So after reorganization the same lines will be used, but some lines will have to transmit electricity in the direction opposite to the current one. To make the reorganization gentle enough, the management of the power network is planning to switch the transmission direction for exactly one line each day. Of course, no day there must be a cycle in a network, since this may cause damage to the system. The resulting network is also designed to be acyclic.

Help them to plan the reorganization.

Input

The first line of the input file contains n — the number of nodes in the network, and m — the number of transmission lines ($2 \le n \le 1\,000$, $1 \le m \le 10\,000$). The following m lines contain three integer numbers each. The first two give the source and the destination node for the corresponding line in the current node. The third number is 0 if the line must keep its transmission direction in the resulting network, and 1 if the direction must be reversed.

There can be several lines connecting the same pair of nodes, but due to acyclicity condition, they all transmit electricity in the same direction. This is also the reason why no line connects a node to itself.

Output

First output k — the number of days in the plan you suggest. You don't need to minimize this number, but it must not exceed $4m$. After that print k integer numbers — for each day output the number of the line that changes the transmission direction this day.

If it is impossible to make the desired reorganization, output -1 instead of k.

Example

electricity.in	electricity.out

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

`2 2`
`1 2 1`
`1 2 1`
	`-1`

Problem C. Numbers to Numbers

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

Numbers in English are written down in the following way (only numbers less then 10^9 are considered). Number $\overline{abc\,def\,ghi}$ is written as “[abc] million [def] thousand [ghi]”. Here “[xyz]” means the written down number $\overline{xyz}$.

In the written down number the part “[abc] million” is omitted if $\overline{abc} = 0$, “[def] thousand” is omitted if $\overline{def} = 0$, and “[ghi]” is omitted if $\overline{ghi} = 0$. If the whole number is equal to 0 it is written down as “zero”. Note that words “million” and “thousand” are singular even if the number of millions or thousands respectively is greater than one.

Numbers under one thousand are written down in the following way. The number $\overline{xyz}$ is written as “[x] hundred and [yz]. Here “[x] hundred and” is omitted if $x = 0$. Note that “hundred” is also always singular.

Numbers under 20 are written down as “zero”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”, “ten”, “eleven”, “twelve”, “thirteen”, “fourteen”, “fifteen”, “sixteen”, “seventeen”, “eighteen”, and “nineteen” respectively. Numbers from 20 to 99 are written down in the following way. Number $\overline{xy}$ is written as “[x0] [y]”, and numbers divisible by ten are written as “twenty”, “thirty”, “forty”, “fifty”, “sixty”, “seventy”, “eighty”, and “ninety” respectively.

For example, number 987 654 312 is written down as “nine hundred and eighty seven million six hundred and fifty four thousand three hundred and twelve”, number 100 000 037 as “one hundred million thirty seven”, number 1 000 as “one thousand”. Note that “one” is never omitted for millions, thousands and hundreds.

Given English text with all numbers written as words, transform it to the text where all numbers are written as numbers. For example, the text “he had one hundred and ninety five dogs” must be tranformed to “he had 195 dogs”. You must perform a transformation in such a way that as many words as possible are transformed to numbers, so, for example, “three thousand” must be transformed to “3000”, not to “3 thousand”.

If there are several way to perform the transformation in such a way, you must do it so that the first number is as great as possible, for example “two thousand thirty two” must be transformed to “2032”, not “2030 2”. If there are still several ways, maximize the second number, and so on.

Input

Input file contains an English text that only contains English letters, punctuation marks (‘,’, ‘.’, ‘!’, ‘?’, ’:’, ’;’, ’(’, ’)’), and blanks (spaces and line feeds).

You must perform the transformation in the way described. Words in numbers to be transformed are separated with blanks only (so, for example, “thirty, three” may only be transformed as “30, 3”, not as “33”).

When performing transformation you must replace characters starting from the first letter of the first word in the number to the last one in the last word with digits.

Input file is not empty and its size does not exceed 20000 bytes.

Output

Output the result of the transformation.

Example

numbers.in

```
From three thousand one hundred and fifty teams selected
from one thousand four hundred and eleven universities
in seventy five countries competing at one hundred and
twenty seven sites and hundreds more competing at
preliminary contests worldwide, seventy three teams
of students competed for bragging rights and prizes
at The Twenty Eighth Annual ACM International Collegiate
Programming Contest World Finals sponsored by IBM on
March Thirty One, Two Thousand Four, and hosted at the
Obecni Dum, Prague by Czech Technical University in Prague.
```

numbers.out

```
From 3150 teams selected
from 1411 universities
in 75 countries competing at 127 sites and hundreds more competing at
preliminary contests worldwide, 73 teams
of students competed for bragging rights and prizes
at The 20 Eighth Annual ACM International Collegiate
Programming Contest World Finals sponsored by IBM on
March 31, 2004, and hosted at the
Obecni Dum, Prague by Czech Technical University in Prague.
```

Problem D. Police Cities

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

Once upon the time there lived a king and he had a big kingdom. And there were n cities in his kingdom and some of them were connected by the roads. And the roads were all one-way because it would be dangerous if two carriages riding in opposite directions met on a road.

And once the king decided that he would like to establish police in his country and ordered to build police stations in some cities. But since his finances are limited, he would only like build police stations in k different cities. He would like to build them in such a way, that the following conditions were satisfied:

- it is possible to get by the roads from each city to some city with the police station;
- it is possible to get by the roads to each city from some city with the police station.

Now the king wants to know how many different ways are there to do so. Help him to find the answer to this question.

Input

The first line of the input file contains n, m and k — the number of cities and roads in the kingdom, and the number of police stations to build, respectively ($1 \le n \le 100$, $0 \le m \le 20\,000$, $1 \le k \le n$). The following m lines contain two city numbers each and describe roads, remember that it is only possible to travel along roads in one direction — from the first city to the second one. Two cities may be connected by more than one road.

Output

Output the only integer number — the number of ways to fulfil king's request.

Example

police.in	police.out

6 7 3
1 2
2 3
3 1
3 4
4 5
5 6
6 5
	15

Problem E. Quadratic Equation

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

Children in school learn how to solve quadratic equations — that is, equations of form

$$ax^2 + bx + c = 0,$$

where a, b and c are some given real numbers, and x is the real number to find.

In this problem you have to solve quadratic equation for polynomials with coefficients from $\mathbb{Z}/2\mathbb{Z}$. Recall, that there are two numbers in $\mathbb{Z}/2\mathbb{Z}$: 0 and 1, and all operations in this field are performed modulo 2.

Given polynomials $a(t)$, $b(t)$ and $c(t)$, find such polynomial $x(t)$ that

$$a(t)x^2(t) + b(t)x(t) + c(t) = 0,$$

where equality should be considered as polynomial equality. Remember, that two polynomials are equal if and only if their coefficients at corresponding powers of t are equal.

Input

Input file contains $a(t)$, $b(t)$ and $c(t)$, specified as their power followed by their coefficients, starting from the leading one (the coefficient at the greatest power of t). Zero polynomial has the degree of -1 for the purpose of this problem. Degrees of all polynomials do not exceed 127.

Output

If there is at least one solution to the equation, output any one in the same format, that is used in input. Leading coefficient of the answer polynomial must not be zero. The degree of the polynomial must not exceed 512.

In the other case print "`no solution`" on the first line of the output file.

Example

quadratic.in	quadratic.out

`0 1`
`2 1 1 0`
`3 1 0 0 0`
	`1 1 0`

`0 1`
`1 1 1`
`0 1`
	`no solution`

`-1` `-1` `-1`	`-1`

In the first example the equation has the form

$$x(t)^2 + (t^2 + t)x(t) + t^3 = 0,$$

and $x(t) = t$ is clearly a solution.

In the second example the equation is

$$x(t)^2 + (t + 1)x(t) + 1 = 0,$$

and there is no solution.

In the third example the equation is $0 = 0$ and any polynomial is a solution.

Problem F. Restore the Tree

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

An undirected connected graph with no cycles is called a *tree*. A vertex with degree equal to one in a tree is called a *leaf*.

Consider a tree. For each pair of leaves one can determine the distance between these leaves — the number of edges one needs to walk to get from one leaf to another. Given these distances for all pairs of leaves, you need to restore the tree.

Input

The first line of the input file contains l — the number of leaves in the tree ($2 \le l \le 200$). Next l lines contain l integer numbers each — distances between leaves. It is guaranteed that no distance exceeds 200, all distances are non-negative, distance from a leaf to itself is zero, and the distance from leaf i to leaf j is equal to the distance from leaf j to leaf i.

Output

If it is impossible to restore the tree because no such tree exists, output -1 on the first line of the output file.

In the other case first output n — the number of vertices in the tree. After that output $n - 1$ pairs of numbers — edges of the tree, each specified with two vertices it connects. If there are several solutions, output any. First l vertices must correspond to tree leaves as they are described in the input file, other vertices may be numbered in arbitrary order.

Example

restore.in	restore.out

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

Problem G. Unrhymable Rhymes

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

An amateur poet Willy is going to write his first abstract poem. Since abstract art does not give much care to the meaning of the poem, Willy is planning to impress listeners with unusual combinations of words. He prepared n lines of the future poem, but suddenly noticed that not all of them rhyme well.

Though abstractionist, Willy strongly respects canons of classic poetry. He is going to write the poem that would consist of quatrains. Each quatrain consists of two pairs of rhymed lines. Therefore there can be four types of quatrains, if we denote rhymed lines with the same letter, these types are "AABB", "ABAB", "ABBA" and "AAAA".

Willy divided the lines he composed into groups, such that in each group any line rhymes with any other one. He assigned a unique integer number to each group and wrote the number of the group it belongs next to each line. Now he wants to drop some lines from the poem, so that it consisted of correctly rhymed quatrains. Of course, he does not want to change the order of the lines.

Help Willy to create the longest poem from his material.

Input

The first line of the input file contains n — the number of lines Willy has composed ($1 \le n \le 4000$). It is followed by n integer numbers denoting the rhyme groups that lines of the poem belong to. All numbers are positive and do not exceed 10^9.

Output

On the first line of the output file print k — the maximal number of quatrains Willy can make. After that print $4k$ numbers — the lines that should form the poem.

Example

`rhymes.in`	`rhymes.out`

`15`
`1 2 3 1 2 1 2 3 3 2 1 1 3 2 2`
	`3`
	`1 2 4 5`
	`7 8 9 10`
	`11 12 14 15`

`3`
`1 2 3`
	`0`

Problem H. Selling Tickets

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

You are hired by a company *Antarctic Railways* for developing the new ticket selling system *Express 3.14*. The problem looks very simple: the only type of cars run by a company is a car that has 36 seats united to 9 four-seat compartments. The system gathers information about passengers that wish to travel together and distributes seats among passengers to satisfy them as much as possible.

In particular, given up to 36 ticket requests for travel, divided into groups of up to four people that wish to travel together, you have to assign seat numbers to passengers in a way to maximize the *total satisfaction*. A total satisfaction is the sum of *satisfactions* of each passenger. A satisfaction of each passenger is the product of the *friendship coefficient* of his group and the number of other passengers from his group that travel in the same compartment. For example, if the group of four with friendship coefficient 30 travels in one compartment, they increase the total satisfaction by $4 \cdot 3 \cdot 30 = 360$, and if they traveled in two compartments as two and two, they would increase the total satisfaction by $2 \cdot 1 \cdot 30 + 2 \cdot 1 \cdot 30 = 120$.

Input

The first line of the input file contains m — the number of groups. Next m lines contain descriptions of groups. Each group is described with the number of passengers in it (one to four), its friendship coefficient (positive integer number not exceeding 1000) and the identifiers of passengers in it. Each passenger is identified with some unique positive integer number not exceeding 100. The total number of passengers does not exceed 36. No passenger is listed in more than one group.

Output

On the first line of the output file print the maximal possible total satisfaction. The next 9 lines must contain four integer numbers each — the identifiers of passengers travelling in the corresponding compartment. If some seat remains unoccupied, output 0 for it.

Example

`tickets.in`	`tickets.out`

```
11
3 30 1 2 3
3 30 4 5 6
3 30 7 8 9
3 30 10 11 12
3 30 13 14 15
3 30 16 17 18
3 30 19 20 21
3 30 22 23 24
3 30 25 26 27
4 10 28 29 30 31
4 10 32 33 34 35
```
	```
	1620
	1 2 3 28
	4 5 6 29
	7 8 9 30
	10 11 12 31
	13 14 15 32
	16 17 18 33
	19 20 21 34
	22 23 24 35
	25 26 27 0
	```

Problem I. Traces

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

In a theory of parallel systems a special generalization of words is used, called *traces*. Consider a finite alphabet Σ and a symmetric relation $\mathcal{I} \subset \Sigma \times \Sigma$, called *independence relation*. Letters that satisfy independence relation are called *independent* or *commuting*.

Consider all finite words over Σ. We say that a word w_1 can be transformed to a word w_2 if we can obtain w_2 from w_1 by sequentially swapping zero or more pairs of adjacent commuting letters in it.

For example, let $\Sigma = \{a, b, c\}$ and $\mathcal{I} = \{(a, b), (b, c)\}$. In this case word "*abbcabc*" can be transformed to word "*abcacbb*" ("*abbcabc*"$\to$"*abcbabc*"$\to$"*abcabbc*" $\to$"*abcabcb*"$\to$"*abcacbb*"), but it cannot be transformed to, say, word "*aabbbcc*".

It is clear, that the relation "*can be transformed to*" is an equivalence relation on words. Equivalence classes for this relation are called *traces*. In other words, a trace is a set of words that some given word can be transformed to, two words belong to the same trace if one can be transformed to another.

Given two words, detect whether they belong to the same trace.

Input

The first line of the input file contains n — the number of letters in the alphabet Σ ($2 \le n \le 10$, let Σ consist of first n small letters of English alphabet), and m — the number of independent pairs of letters. Next m lines contain two letters each and define independence relation. The last two lines contain two words to check, each not longer than 100 000 characters.

Output

Output "`YES`" if two given words belong to the same trace and "`NO`" if they do not.

Example

traces.in	traces.out

`3 2`
`ab`
`bc`
`abbcabc`
`abcacbb`
	`YES`

`3 2`
`ab`
`bc`
`abbcabc`
`aabbbcc`
	`NO`

Problem J. Ray Tracing

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

Johnie is preparing a new commercial for *Caca Cola* company producing a well known refreshing drink. As a part of the commercial there must be a can of *Caca Cola* drink flying all around the screen.

The whole commercial is supposed to be rendered on a computer using backward ray tracing technology. Since the can is a cylinder, Johnie now has to solve a problem of intersecting a ray and a cylinder. Help him!

Given a cylinder and a ray, find the point of their intersection, closest to the origin of the ray.

Input

Input data specify the cylinder and the ray. Cylinder is specified with two points and its radius. The two points are the centers of circles that bound the cylinder.

The ray is specified with its origin point and any point on it, being at the distance of at least 0.1 from the origin.

All numbers in the input file do not exceed 10^3 by their absolute value, cylinder radius and height (distance between centers of the circular sides of the cylinder) are at least 0.01. The origin of the ray is outside the cylinder.

Output

Output the coordinates of the point of intersection of the cylinder and the ray, closest to the origin of the ray. If the ray does not intersect the cylinder, output the word "`NONE`" instead. It is guaranteed that the ray does not touch the cylinder.

Print at least four digits after the decimal point.

Example

tracing.in	tracing.out

`0.0 0.0 0.0`
`0.0 0.0 2.0`
`1.5`
`4.0 4.0 1.0`
`2.0 2.0 1.0`
	`1.06066017 1.06066017 1.00000000`

`0.0 0.0 0.0`
`0.0 0.0 2.0`
`1.5`
`4.0 4.0 1.0`
`2.0 2.0 2.0`
	`NONE`

`1.0 1.7 0.3`
`-4.2 0.9 -0.1`
`1.2`
`0.7 0.3 -2.0`
`2.7 2.3 5.0`
	`1.15122867 0.75122867 -0.42069966`