Problem A. Addictions

Input file:	`addict.in`
Output file:	`addict.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

Many people have addictions. Some of them are quite harmless (or at least considered to be such), others may be really dangerous for your health or even life.

Overcoming your addictions is often hard, because of so called *withdrawal syndrome*. When you remove some of your addictions, you start missing it very much. The stronger is the addiction the harder is to overcome the withdrawal syndrome. Sometimes to overcome it, people get other addictions that can partially compensate the missing one.

Josh is planning to remove most of his addictions. He has classified them and for each of his addictions he knows its strength s_i.

He decided to remove one addiction per week. But since giving addictions up is hard, sometimes he must simultaneously restore some of his former addictions. Josh has inspected his will-power and decided that he can deal with his problems if after removing the addiction and restoring some others (possibly none), the summary strength of the addictions decreases by no more than d. Of course, Josh doesn't want to risk his health, so he doesn't change his addictions in such a way, that their summary strength increases.

Help Josh to remove the maximal number of addictions and do it in a least number of weeks.

Input

The first line of the input file contains n — the number of addictions Josh has ($1 \le n \le 1000$) and d ($1 \le d \le 10000$). The following n lines describe addictions — each line contains the addiction name (a string of at most 40 English letters) and its strength (positive integer not exceeding 10000).

Output

At the first line of the output file print k — the maximal number addictions Josh can remove and t — the minimal number of weeks needed to do so. The following k lines must contain addictions he can remove, one on a line, sorted in alphabetical order.

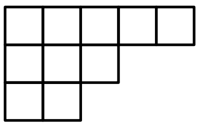
Examples

addict.in	addict.out
5 3 Girls 7 Coffee 1 Drinking 4 Smoking 5 Contests 25	4 9 Coffee Drinking Girls Smoking

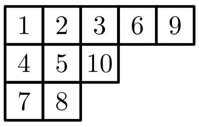
Problem B. Chess Tableaux

Input file:	`chess.in`
Output file:	`chess.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

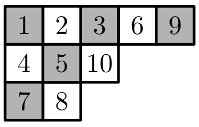
Consider some partition of an integer number $n = m_1 + m_2 + \ldots + m_k$ where $m_1 \geq m_2 \geq \ldots \geq m_k$. This partition may be illustrated by the *Young diagram* — n boxes arranged in k rows, where k is the number of terms in the partition. A row representing the number m_i contains m_i boxes. All rows are left-aligned, and sorted from longest to shortest.



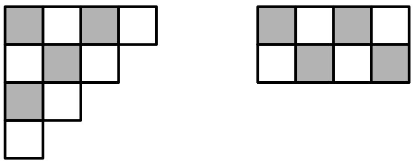
Young diagram can be converted to *Young tableau* by putting integer numbers from 1 to n to boxes, in such a way that the numbers in each row and each column increase from left to right, and from top to bottom, respectively.



The Young tableau is called a *chess tableau* is after coloring its boxes as on a chessboard (top left box colored black), black boxes contain odd numbers, and white boxes contain even numbers.



Clearly, not every Young diagram can be converted to a chess tableau. For example, none of the two diagrams below can be converted to a chess tableau.



Given n, find the number of partitions of n such that the corresponding Young diagram can be coverted to a chess tableau.

Input

Input file contains n ($1 \leq n \leq 50$).

Output

Output one integer number — the number of partitions of n for which there is a chess tableau.

Examples

chess.in	chess.out
10	36

Problem C. Decoding Martian Messages

Input file:	`decoding.in`
Output file:	`decoding.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

Decoding messages sent over channels with noise is always a hard problem. Various methods are used to detect transmission errors and fix them. Peter is working in a company that develops the programming environment for the communicating system that will be installed on a new Mars exploring module.

The messages used in a communication can be considered as the sequences of letters of the English alphabet. To help detecting and fixing errors, the following condition is satisfied for all transmitted messages: each k consecutive letters from the message form the word that belongs to the given set of allowed words D.

For example, if $k = 2$ and $D = \{$“`ab`”, “`ba`”, “`aa`”$\}$, the word “`abaab`” can be transmitted, but the word “`abbab`” cannot.

When the message is received, it is difficult to decode each character separately because of a random noice added to the signal. For each position i of the message and each character c the probability $p_{i,c}$ of this character occuring at this position is known. The decoder must find the most probable message that was transmitted, that is — the valid message $m_1 m_2 \ldots m_n$ such that the product $\prod_{i=1}^{n} p_{i,m_i}$ is maximal possible.

Writing such decoder is exactly your task.

Input

The first line of the input file contains d, k and t — the number of words in D, the length of all those words and the number of characters that can be used in the words, respetively ($1 \le d \le 200$, $1 \le k \le 10$, $2 \le t \le 26$). The following d lines contain one word each. All words contain only first t small letters of the English alphabet.

The next line contains n — the length of the message ($k \le n \le 1000$). The following n lines contain t real numbers each — the j-th number in the i-th of these lines is the probability that the i-th character of the message is equal to the j-th character of the alphabet. The sum of the probabilities in each line is equal to 1.

Output

If no message of length n exists such that all of its subwords belong to D, or no word has a positive probability, print “`---`” (three minuses) to the output file. In the other case output the most probable message. If there are several solutions, print any one.

Examples

decoding.in	decoding.out

```
3 2 2
ab
ba
aa
5
0.8 0.2
0.2 0.8
0.2 0.8
0.8 0.2
0.2 0.8
```
	`abaab`

Problem D. Gas Problem

Input file:	`gas.in`
Output file:	`gas.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

Flatland's neighbour Edgeland has decided to stop importing gas from Flatland because its price went way too high. After the decision Flatland's gas company stopped pumping gas into the pipe system of Edgeland.

The pipe system of Edgeland consists of a number of gas stations connected by pipes. For each pipe it is known in which direction the gas should be pumped along the pipe.

Unfortunately, there is some problem. It is not good to completely stop gas circulation in pipes. If no gas is pumped through the pipe, its monitoring and controlling systems may get corrupted. For each pipe its minimal gas transit is known. The minimal gas transit of the pipe is the minimal amount of gas that needs to be pumped along this pipe each day to keep it intact.

After some discussion, Edgeland gas company managers decided to use some technical gas to keep the gas system working. The technical gas will be circulating inside the pipes. For each gas station the amount of gas coming to it each day must be equal to the amount of gas pupmped away from it along the pipes. The total amount of gas needed for the project is the sum of amounts of gas needed for each pipe each day.

Given the map of the gas system, help the gas company to find out what minimal amount of gas is needed to keep the gas system intact.

Input

The first line of the input file contains n and m — the number of gas stations and gas pipes in the gas system of Edgeland ($2 \le n \le 300$, $2 \le m \le 1000$). The following m lines describe gas pipes, each pipe is specified with three integer numbers — the number of the gas station where the gas must be pumped into the pipe, the number of the gas station where the gas is going along this pipe, and the minimal gas transit of the pipe. The minimal gas transit of the pipe is integer and does not exceed 10^3.

There is at least one pipe leaving from each gas station, and at least one pipe entering each gas station. No two stations are connected by more than one pipe, if there is a pipe between two stations, there is no pipe in the reverse direction.

Output

At the first line of the output file print the minimal amount of gas needed to keep the gas system intact. Each of the following m lines must contain the amount of gas that will be transported along the corresponding pipe each day.

If there is no way to keep the gas system intact, print "-1" at the first line of the output file.

Examples

gas.in	gas.out

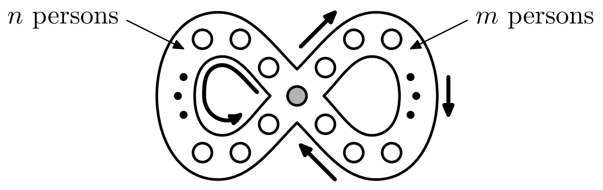
4 5
1 2 1
2 3 1
1 3 1
4 1 3
3 4 3
	10
	1
	1
	2
	3
	3

Problem E. Infinity Sect

Input file:	`infinity.in`
Output file:	`infinity.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

The members of the *Infinity Sect* worship infinity. Joseph is a mathematician, and he is very amenable. So the leaders of the sect managed to get Joseph in.

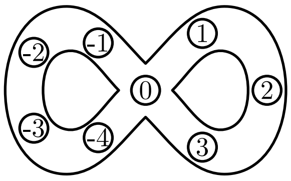
Once a year there is a selection of the leader of the sect. Of course, the procedure is closely connected to the infinity. All $m + n + 1$ members of the sect stand forming the infinity sign, as shown on the picture.



The k-th person starting from the one in the center (gray dot on a picture above) is counted. First the persons in the part of the infinity sign that contains m persons are counted, then the persons that are in the n-part (arrows on the picture shows the order of counting). The k-th person goes away from the sign, and counting is continued from the next person. Again, the k-th one is counted and goes away. The process continues until only one person is left. The person left becomes the leader of the sect for the next year.

Note that when counting, the person that is standing in the center is counted twice when going around the sign — before the m-part and before the n-part.

Let us number the persons in the following way: the person in the center has number 0, persons in the m-part have numbers from 1 to m in order they are counted, persons in the n-part have numbers from -1 to $-n$ in order they are counted.



For example, let $m = 3$, $n = 4$, $k = 6$. The counting proceeds as follows: $0, 1, 2, 3, 0, -1$, and the person -1 goes away. Then the counting proceeds: $-2, -3, -4, 0, 1, 2$, and the person 2 goes away. After that 0 goes away, followed by $1, -2, 3$ and -4. The person -3 is left — he is the leader.

Of course, Joseph would like to become the leader of the sect. Help him to find which position to occupy at the sign to do it.

Input

The input file contains n, m and k ($1 \le n$, $1 \le m$, $n + m \le 500\,000$, $1 \le k \le 10^9$).

Output

Print the position taht Joseph must occupy to the output file.

Examples

`infinity.in`	`infinity.out`
`3 4 6`	`-3`

Problem F. Polymorphism

Input file:	`polymorphism.in`
Output file:	`polymorphism.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

Most modern programming languages allow polymorphism — using the same name for different routines. The compiler decides which function or method to call depending on the current context. Let us describe simple function-oriented language that supports namespaces.

The program in this language is a sequence of functions and namespaces. Each namespace may in turn contain functions and namespaces. A function is a sequence statements. We will only consider statements that are function calls.

Each namespace and each function has a *name*. Let us define the *fully-qualified* name of a namespace (function) recursively. For the top-level namespace (function) its fully-qualified name coincides with its name. In the other case, if the corresponding namespace (function) "`s`" belong to the namespace with the fully-qualified name "`n`", the fully-qualified name of this namespace (function) is "`n.s`".

If the statement "`f`" in a function located in a namespace with the fully quialified name "`n`" is called, the compiler first tries to create the call for the function with the fully-qualified name "`n.f`". If no such function exists, the compiler must consider the namespace "`m`" that the namespace "`n`" belongs to, and try to create the call for the function "`m.f`", if no such function exists, the parent namespace for "`m`" must be considered, and so on. If no suitable function is found, error is generated.

Given the program in a language described, you must enumerate all of its functions and for each function call you must specify which function will be called.

Input

The input file contains the program in a language described.

The grammar of the language is the following. Here $\langle\text{expression}\rangle^*$ means that $\langle\text{expression}\rangle$ may occur zero or more times.

$$\begin{aligned}
\langle\text{program}\rangle &\longrightarrow \varepsilon \mid \langle\text{namespace}\rangle\,\langle\text{program}\rangle \mid \langle\text{function}\rangle\,\langle\text{program}\rangle \\
\langle\text{namespace}\rangle &\longrightarrow \textit{namespace}\,\langle\text{id}\rangle\,\{\langle\text{program}\rangle\} \\
\langle\text{function}\rangle &\longrightarrow \textit{function}\,\langle\text{id}\rangle\,\{\langle\text{body}\rangle\} \\
\langle\text{body}\rangle &\longrightarrow \varepsilon \mid \langle\text{statement}\rangle\,;\,\langle\text{body}\rangle \\
\langle\text{statement}\rangle &\longrightarrow \langle\text{id}\rangle \mid \langle\text{id}\rangle\,.\,\langle\text{statement}\rangle \\
\langle\text{id}\rangle &\longrightarrow \langle\text{char}\rangle\,\langle\text{char}\rangle^* \\
\langle\text{char}\rangle &\longrightarrow A \mid B \mid \ldots \mid Z \mid a \mid b \mid \ldots \mid z
\end{aligned}$$

The program is guaranteed to be syntactically correct. No namespace, including the unnamed top-level one, contains two namespaces with the same name, two functions with the same name or a function and a namespace with the same name.

All names and keywords are case sensitive.

There can be whitespaces between tokens, they should be ignored. There is always at least one whitespace between "`namespace`" keyword and namespace name, and "`function`" keyword and function name.

There are at most 100 different namespaces, at most 2000 functions, the length of all names doesn't exceed 20. The size of the input file does not exceed 100 kilobytes.

Output

For each function in the input program print its descrption.

The description must start with its number in square brackets followed by the fully-qualified function name. Enumerate functions starting from 1 in order of appearance. For each statements in a function print two spaces followed by the number of the function called in this statement in round brackets. If no such function exists, print “(ERROR)” instead.

Functions must be described in the order they are declared in the input file.

Examples

polymorphism.in

```
namespace a {
  function y {
    y;
  }

  function z {
    y;
  }

  namespace b {
    function x {
      y;
      a.y;
    }
    function y {
      x;
      y;
      z;
    }
  }
}

function x {
  x;
  a.x;
  a.b.x;
  y;
  a.y;
  a.b.y;
  z;
  a.z;
  a.b.z;
}
```

polymorphism.out

```
[1] a.y
  (1)
[2] a.z
  (1)
[3] a.b.x
  (4)
  (1)
[4] a.b.y
  (3)
  (4)
  (2)
[5] x
  (5)
  (ERROR)
  (3)
  (ERROR)
  (1)
  (4)
  (ERROR)
  (2)
  (ERROR)
```

Problem G. Refrain

Input file:	`refrain.in`
Output file:	`refrain.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

Andrew is writing the new plugin for *Windump* media player. The goal of the plugin is to detect refrain in various songs.

Since it is difficult to exactly tell what the refrain is, Andrew has developed the following mathematical model. The song is considered as the sequence of the notes. The segment of a sequence is one or more consecutive notes in it. The *repetition value* of a segment is the length of the segment in notes multiplied by the number of times it occurs in the sequence. For example, the repetition value of the segment "$1, 2, 1$" in the sequence "$1, 2, 1, 2, 1, 3, 1, 2, 1$" is 9 since its length is 3 and it occurs 3 times in the sequence.

The segment of the sequence is considered a refrain, if its repetition value is maximal among all segments of the sequence.

Help Andrew to write the plugin — given sequence of notes, find the refrain in it.

Input

The first line of the input file contains n and m — the number of notes in a sequence and the number of various notes used in a song ($1 \le n \le 150\,000$, $1 \le m \le 10$). The second line contains n integer numbers ranging from 1 to m — the given sequence of notes.

Output

At the first line of the output file print the repetition value of the refrain. At the second line print the length of the refrain. At the third line print the refrain itslef. If there are several potential refrains, output any one.

Examples

refrain.in	refrain.out

9 3
1 2 1 2 1 3 1 2 1
	9
	3
	1 2 1

Problem H. Star Polygon

Input file:	`star.in`
Output file:	`star.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

The new secret military polygon is planned in Starland. The polygon with the code name *Star Polygon* will be used for demonstrating the newest military technology achievements to the head officers of the army.

Of course, all technologies demonstrated on the polygon are top secret, so the polygon will be surrounded by a high wall. The wall will have a form of a simple closed polyline, the guard towers will be built in its vertices.

Since the head officers would like to see everything that is going on at the polygon during the demonstration, the special watchtower will be built inside the polygon. The watchtower must be located in such a point, that all points inside the polygon including all inner points of its wall, are visible from it. If the line of sight of an officer goes along the side, he is not able to see it.

The construction company that won the governmental tender for the construction of the polygon, has built the guard towers at the boundary of the polygon, and now is planning to connect all of them by the wall. Of course, they want to build as short wall as possible. However, it suddenly turned out that some polylines with vertices in the guard towers are not suitable for the polygon. That is — in some cases it is impossible to build the watchtower inside the polygon so that its whole area and boundary are visible from it.

Now the managers of the company wonder — what is the shortest closed polyline with vertices at the guard towers already built, such that there is a point inside it where the watchtower can be built. Help them to find that out.

Input

The first line of the input file contains n — the number of the guard towers built ($3 \le n \le 15$). The following n lines contain two integer numbers each — the coordinates of the guard towers. Coordinates do not exceed 1000 by their absolute values. There are three points that do not belong to the same line.

Output

Print the length of the shortest possible wall at the first line of the output file. The second line must contain n integer numbers — the numbers of the guard towers in order they should appear along the wall in the counterclockwise order. The third line must contain two real numbers — the coordinates of the watchtower.

Your answer must be accurate up to 10^{-6}.

Examples

star.in	star.out

5
0 0
1 1
2 2
0 2
2 0
	8.8284271247461901
	1 5 3 2 4
	1.0 0.5

Problem I. Superinvolutions

Input file:	`superinv.in`
Output file:	`superinv.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

Recall that the permutation $\alpha = \langle a_1, a_2, \dots a_n\rangle$ is called an *involution* if $a_{a_i} = i$ for each i.

The permutation $\beta = \langle b_1, b_2, \dots b_k\rangle$ is called the *subpermutation* of the permutation $\alpha = \langle a_1, a_2, \dots a_n\rangle$ if there is a sequence $i_1 < i_2 < \dots < i_k$ such that $b_j = a_{i_j}$ for each j.

Given n and the permutation $\beta = \langle b_1, b_2, \dots b_k\rangle$, find the number of involutions that contains β as a subpermutation.

Input

The first line of the input file contains n and k ($1 \le k \le n \le 200$). The second line contains the permutation β of length k.

Output

Output one integer number — the number of involutions of length n that contain β as a subpermutation.

Examples

superinv.in	superinv.out
5 3 3 1 2	3

In the example the corresponding involutions are $\langle 3, 4, 1, 2, 5\rangle$, $\langle 3, 5, 1, 4, 2\rangle$, $\langle 4, 5, 3, 1, 2\rangle$.

Problem J. Truth

Input file:	`truth.in`
Output file:	`truth.out`
Time limit:	4 seconds
Memory limit:	64 megabytes

The problem of deciding whether the boolean formula is satisfiable — that is, whether it is possible to assign such values to the variables that it is evaluated to 1 — is a well known $\mathcal{NP}$-complete problem. It is widely believed that this problem doesn't have the polynomial solution.

In this problem you will be given the alternative problem. You are given a boolean formula that contains constants '`0`' and '`1`', negation '`!`', conjunction '`&`' and disjunction '`|`' (listed from highest priority to lowest). The formula doesn't contain parenthesis. You have to decide whether it is possible to add parenthesis to the formula so that it became true.

Input

The input file contains the boolean formula that doesn't contain parenthesis. The length of the formula doesn't exceed 500 characters.

Output

If it is impossible to add parenthesis to the formula so that it became true, print "`Impossible`" at the first line of the output file. In the other case print the resulting formula.

Examples

truth.in	truth.out
`1|1&0`	`1|(1&0)`
`0&0`	`Impossible`
`!0&0`	`!(0&0)`