Problem A. Cookies

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

Santa Claus is planning to bring gifts to n children. He has m cookies and is planning to divide them to n piles. However, as usually problems come unexpected. The child gets unhappy if somebody gets more cookies than him.

Each child is characterized by his *greediness*, the greediness of the i-th child is g_i. The unhappines of the i-th child is equal to $g_i a_i$ where a_i is the number of children that get more cookies than him.

Now Santa wants to divide cookies in such a way that the total unhappiness is minimized. Each child must get at least one cookie. Santa would like to give away all m cookies he has. Help him to do so.

Input

The first line of the input file contains n and m ($1 \le n \le 30$, $n \le m \le 5000$). The second line contains n integer numbers $g_1, g_2, \ldots, g_n$ ($1 \le g_i \le 10^7$).

Output

Print the minimal possible unhappiness at the first line of the output file. The second line must contain n integer numbers — the number of cookies the corresponding child must get. If there are several solutions, output any one.

Example

cookies.in	cookies.out
3 20 1 2 3	2 2 9 9
4 9 2 1 5 8	7 2 1 3 3

Problem B. Ear Decomposition

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

Consider a connected undirected graph with no loops or parallel edges. Let us call the number of edges incident to vertex v its *degree* and denote it as $\deg v$.

A simple path $v_0 - v_1 - \ldots - v_k$ such that $\deg v_0 \geq 2$, $\deg v_k \geq 2$, and for all i from 1 to $k-1$ $\deg v_i = 2$, is called an *ear*. In particular, if $k = 1$, an edge $v_0 - v_1$ connecting two vertices of degree at least 2 is also an ear. The vertices v_0 and v_k can coincide.

Let us consider an ear $v_0 - v_1 - \ldots - v_k$ and remove all of its edges and intermediate vertices $(v_1, v_2, \ldots, v_{k-1})$ from the graph. This operation is called *ear cut*. If the ear has no intermediate vertices, it can be cut by simply removing its edge.

The *ear decomposition* of the graph G is the sequence of ear cuts, such that after each cut the graph remains connected, and after the last cut the remaining graph consists of a single vertex.

Given graph G find its ear decomposition or report that it doesn't have one.

Input

The first line of the input file contains n and m — the number of vertices and edges of G respectively ($2 \leq n \leq 20\,000$, $n - 1 \leq m \leq 100\,000$). Let the vertices be numbered from 1 to n. The following m lines describe graph edges, each line contains two integer numbers — the numbers of vertices connected by the corresponding edge. It is guaranteed that G is connected. No two vertices of G are connected by more than one edge. No edge connects a vertex to itself.

Output

The first line of the output file must contain d — the number of cuts in the ear decomposition of G. The following d lines must describe cuts. Each cut must be described by k — the number of edges in the corresponding ear, followed by $k + 1$ numbers — the vertices of the ear as they appear along it.

If there is no ear decomposition of the given graph, output $d = -1$.

Example

ear.in	ear.out
5 8 1 2 2 3 3 5 5 1 2 4 4 1 3 4 4 5	4 1 2 3 2 4 3 5 2 4 5 1 3 4 1 2 4
3 2 1 2 1 3	-1

Problem C. ePig

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

Andrew and Ann are developing the new P2P software network *ePig*. The network is intended to be used for sharing files. In this problem you will have to simulate the operation of the network when distributing one large file.

Let there be n clients numbered from 1 to n. Initially the whole file is provided by client 1. All other clients wish to get this file. The file is split into k chunks numbered from 1 to k. The transfer consists of a series of rounds. Each round takes one minute and the bandwidth of the connection of each client is enough to transform one chunk to the client and transform one chunk away from the client (simultaneously). After a client gets some chunk it starts to provide it to other clients.

Before a round each client decides which chunk it will request. The client will request the chunk that is provided by the smallest number of clients (except those chunks that it already has). If there are several such chunks, it selects the one which has the smaller number.

After that the clients make chunk requests. Each client selects the client which has the chunk that it decided to request, if there are several such clients, the client which provides the smallest number of chunks is selected. If there are still several possible variants, the client which has the smallest number is selected.

Each client considers all requests and satisfies one of them. The request satisfied by client X is the one which comes from the most valued client. The value of the client is the number of chunks it allowed X to be downloaded from him in the past. If there are several equally valued clients, X gives the chunk to the one which has the smallest number of chunks available. If there are still several possible variants, the chunk is provided to the client which has the smallest number.

After the requests that will be satisfied are selected the round begins. The clients whose requests were rejected do not download anything this round, all other clients download the chunk they requested. After that the new round starts.

Given n and k, you have to find for each client, what is the number of rounds before it gets the whole file.

Input

The first line of the input file contains n and k ($2 \le n \le 100$, $1 \le k \le 200$).

Output

Output $n - 1$ numbers — for each client except the first one print the number of rounds before it gets the whole file.

Example

epig.in	epig.out
3 2	3 3

The file distribution will proceed as follows. At the first round clients 2 and 3 will request chunk 1 from client 1. Request from client 2 will be satisfied. After that clients 2 and 3 will request chunk 2 from client 1. Client 3 will be satisfied. On the third round client 2 will request chunk 2 from client 3 and client 3 will request chunk 1 from client 2, both requests will be satisfied, and both will have the whole file.

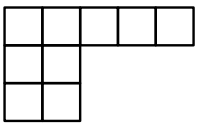
Problem D. Irreducible Young Diagrams

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

Young diagram is a well known way to describe a partition of a positive integer number. A partition of a number n is a representation as a sum of one or several integer numbers $n = m_1 + m_2 + \ldots + m_k$ where $m_1 \geq m_2 \geq \ldots \geq m_k$.

A diagram consists of 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.

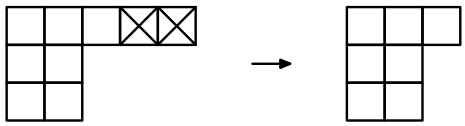
The diagram on the picture below corresponds to the partition $9 = 5 + 2 + 2$.



Let us describe a way to transform the Young diagram. You are allowed to choose two adjacent boxes in the diagram and remove them. The only restriction is that after the process the remaining diagram must be a valid Young diagram.

For example, removing the last boxes of the second and the third row from the diagram above, we get the diagram for the partition $7 = 5 + 1 + 1$.

Removing the last two boxes of the first row from the same diagram we get the diagram for the partition $7 = 3 + 2 + 2$.



There are not other valid ways to transform the given Young diagram.

The diagram that cannot be transformed in the above way is called *irreducible*. Clearly, an empty diagram is irreducible.

Each diagram can be transformed until it is irreducible. Generally there may be several possible ways of transforming the diagram. You are given a Young diagram. You have to find all possible irreducible diagrams that the given one can be transformed to.

Input

The first line of the input file contains k — the number of rows in the diagram ($1 \leq k \leq 100\,000$). The second line contains k numbers: $m_1, m_2, \ldots, m_k$. The sum $n = m_1 + m_2 + \ldots + m_k$ doesn't exceed 10^8.

Output

The first line of the output file must contain one number l — the number of irreducible diagrams the given diagram can be transformed to. The following l lines must describe these diagrams: each line must contain t — the number of rows in the corresponding diagram, followed by t numbers — the number of boxes in corresponding rows.

Example

irreducible.in	irreducible.out
3 5 2 2	1 1 1
1 2	1 0

Problem E. Permutation Reconstruction

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

The famous Ulam Conjecture claims that if we take a graph G and consider a multiset G_{min} of graphs that are obtained from G by removing one of its vertices and all edges incident to it, then the graph G can be reconstructed from G_{min}. The Ulam conjecture is still not proven and no counterexample is known.

In this problem we will consider a similar problem for permutations. Consider a permutation $a = \langle a_1, a_2, \dots a_n \rangle$ of numbers from 1 to n. Let us denote as a/i the permutation of $n-1$ numbers obtained from a by removing a number i and decreasing all numbers greater than i by one.

For example, if $a = \langle 1, 3, 5, 2, 6, 4 \rangle$ then $a/2 = \langle 1, 2, 4, 5, 3 \rangle$.

You are given $a/1, a/2, \dots, a/n$ in some arbitrary order. You must restore the original permutation a.

Input

The first line of the input file contains n — the order of the initial permutation ($5 \le n \le 300$). The following n lines contain $n-1$ numbers each and specify a/i for all i in some order.

Output

Output n integer numbers — the permutation a. It is guaranteed that such permutation exists.

Example

permutation.in	permutation.out
6 1 3 5 2 4 1 3 4 2 5 1 2 4 5 3 2 4 1 5 3 1 4 2 5 3 1 3 2 5 4	1 3 5 2 6 4

In this example the factors are given in the following order: $a/6$, $a/4$, $a/2$, $a/1$, $a/3$ and $a/5$.

Problem F. Decoding Prefix Codes

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

A code is a mapping $c : \Sigma \to \Gamma^*$ of characters of the given alphabet Σ to words of some another alphabet Γ. In this problem Σ consists of the first 10 lowercase letters of the English alphabet, and $\Gamma = \{0, 1\}$.

The code is called *prefix-free* or simply *prefix*, if no code word is a prefix of another word. For example, the code $c(\text{'a'}) = 00$, $c(\text{'b'}) = 01$, $c(\text{'c'}) = 1$ is prefix, but the code $c(\text{'a'}) = 0$, $c(\text{'b'}) = 01$ is not.

You are given a text and a string that is obtained from it by encoding it with some prefix code. You must restore the code that was used to encode the text. If there are several possible variants, output any one.

Input

The first line of the input file contains the given text. Its length does not exceed 1000. It contains only letters '`a`'–'`z`' of the English alphabet. There are at most 10 different characters in the given text.

The second line contains the encoded version of the given text. It is guaranteed that it is obtained from the given text by encoding by some prefix code. No word in the code used has length exceeding 10.

Output

Output the prefix code that could be used to encode the text to get the given encoded version. The number of lines must be equal to the number of different characters in the given text. Each line must contain a character followed by a space and the code word for this character.

Example

prefix.in	prefix.out
hello 0100111000	e 001 h 01 l 1 o 000

Problem G. Hungry Queen

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

Consider an infinite chessboard with cells identified by pairs of integer numbers: (x, y). The black queen is initially located at the cell $(0, 0)$. The queen can move horizontally, vertically or diagonally, but cannot move downwards. That is, after each turn the y-coordinate of the cell where the queen is after the turn must be greater or equal to the y-coordinate of the cell where it was before the turn.

There are n white pawns on the board, they are located at cells (x_i, y_i), where $y_i > 0$.

The queen wants to take as many pawns as possible. White pawns do not move, and the queen can make as many consecutive turns as needed. However, each turn the queen must take a pawn.

Find out what is the maximal number of pawns the queen can take, and which pawns it must take to achieve this number.

Input

The first line of the input file contains n ($1 \le n \le 50\,000$). The following n lines contain two integer numbers each — the coordinates (x_i, y_i) of the pawns ($|x_i| \le 10^9$, $0 < y_i \le 10^9$). No two pawns occupy the same position.

Output

The first line of the output file must contain one integer number k — the number of pawns the queens can take. The second line must contain k integer numbers — the numbers of the pawns the queen can take in order she must do it. The pawns are numbered starting from 1 in order they are given in the input file.

Example

queen.in	queen.out
4 1 1 4 3 -1 3 4 2	3 1 3 2

Problem H. Boat Race

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

Jerry is planning to take part in a boat race. The race will take place in a long narrow canal. The canal runs from east to west, the banks of the canal have a form of a polyline.

Let us introduce the coordinate system in such a way that the western end of the canal has x-coordinate equal to 0, the eastern end of the canal has x-coordinate equal to l. The polyline describing the southern bank of the canal has vertices at points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ where $0 = x_0 < x_1 < \ldots < x_n = l$. The northern bank has the same form, but is w units to the north, so it is described by a polyline with coordinates $(x_0, y_0 + w), (x_1, y_1 + w), \ldots, (x_n, y_n + w)$.

Jerry's boat can start the race at any point of a start line (a segment $(0, y_0) - (0, y_0 + w)$) and end the race at any point of the finish line (a segment $(l, y_n) - (l, y_n + w)$). The boat is so small, that it can be considered a point. When moving a boat can "touch" the banks of the canal, moving just along them.

To increase his chances of wining, Jerry wants to know what is the shortest path from the start line to the finish line.

Input

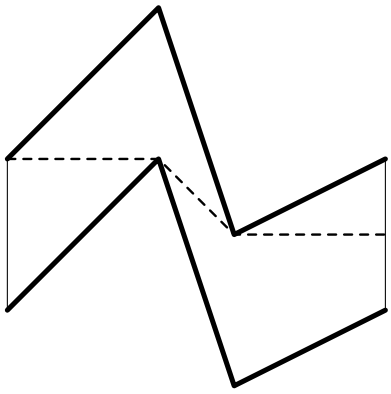
The first line of the input file contains n ($1 \le n \le 100$). The following $n + 1$ lines contain a pair of integer numbers (x_i, y_i) each and describe the southern bank of the canal ($0 = x_0 < x_1 < \ldots < x_n \le 10^4$, $|y_i| \le 10^4$). The last line of the input file contains integer number w ($1 \le w \le 10^4$).

Output

Output one floating point number — the length of the shortest path through the canal from the start line to the finish line. Your answer must be accurate up to 10^{-6}.

Example

race.in	race.out
3 0 0 2 2 3 -1 5 0 2	5.41421356237309505



Problem I. Longest Common Subpair

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

A pair of strings (α, β) is called a *subpair* of a string γ if $\gamma = \gamma_1\alpha\gamma_2\beta\gamma_3$ for some (possibly empty) strings γ_1, γ_2 and γ_3. The length of the pair is the sum of lengths of its strings: $|(\alpha, \beta)| = |\alpha| + |\beta|$.

Given two strings ξ and η find their longest common subpair, that is — such pair (α, β) that it is a subpair of both ξ and η and its length is greatest possible.

Input

Input file contains two strings ξ and η, one on a line. Both strings contain only small letters of the English alphabet. Both string are not empty. The length of each string doesn't exceed 3000.

Output

Output α on the first line of the output file and β on the second line.

Example

subpair.in	subpair.out
abacabadabacaba acabacadacabaca	acaba abaca
ab bc	b

Problem J. New Year Tree Transportation

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

People of Byteland celebrate New Year. Unlike people of most other countries, they do not decorate fir-trees for the New Year celebration. Instead they decorate binary trees. A binary tree is a rooted tree such that every node has at most two children.

A nice binary tree with n nodes was prepared to be set up on the main square of Byteland capital. However first it must be transported from the place where it was grown up to the capital. The transportation will be arranged by the railroad. But it turned out that the standard railroad car can carry the tree only if it has at most k nodes.

So it was decided to cut several edges of the tree so that each of the remaining connected parts had at most k nodes. After the tree is transported to the capital it would be reassembled and set up. Due to security reasons each car must carry only one tree part.

Of course the department of transportation of Byteland would like to use as few cars as possible to transport the tree. However, minimizing the number of cars seemed to be too difficult problem. Therefore the minister of transportation ordered to cut the tree in such a way that the number of cars needed at least did not exceed $\lceil 2n/k \rceil$.

But the people who are transporting the tree couldn't solve even this problem. Help them! Given a binary tree find the way to cut some of its edges in such a way that each of the remaining connected parts had at most k nodes and the number of parts didn't exceed $\lceil 2n/k \rceil$.

Input

The first line of the input file contains n — the number of nodes of the tree, and k — the maximal capacity of the car ($1 \le n \le 100\,000$, $1 \le k \le n$). The following $n-1$ lines describe the edges of the tree. Each edge is described by two integer numbers: the parent node and the child node. The given tree is guaranteed to be a binary tree. Nodes are numbered from 1 to n, root has number 1.

Output

The first line of the output file must contain l — the number of edges that must be cut. The second line must contain l integer numbers — the edges to be cut. Edges are numbered from 1 to $n-1$ as they are listed in the input file.

If the tree cannot be cut in the described way, output $l = -1$.

Example

tree.in	tree.out
5 2 1 2 1 5 5 3 5 4	2 2 4