Problem A. Avangard Latin Squares

Input file:	`avangard.in`
Output file:	`avangard.out`
Time limit:	1 second
Memory limit:	256 megabytes

Latin square of order n is a table with n rows and n columns. Each cell contains integer ranging from 1 to n, so that each row contains each number exactly once and each column contains each number exactly once as well.

The picture below shows Latin square of order 4.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 4 & 3 & 2 & 1 \\ 3 & 1 & 4 & 2 \end{pmatrix}$$

Let us index cells as (r, c) where r is row number couting from top (starting from 1) and c is column number couting from left (starting from 1). For example, cell $(2, 3)$ in the Latin square above contains 1.

Let us call two cells (r_1, c_1) and (r_2, c_2) belonging to the same *primary extended diagonal* if $(r_1 - c_1) - (r_2 - c_2)$ is divisible by n. Namely, 4×4 square contains 4 primary extended diagonals: $\{(1,1),(2,2),(3,3),(4,4)\}$, $\{(1,2),(2,3),(3,4),(4,1)\}$, $\{(1,3),(2,4),(3,1),(4,2)\}$, and $\{(1,4),(2,1),(3,2),(4,3)\}$.

Similarly, two cells (r_1, c_1) and (r_2, c_2) belong to the same *secondary extended diagonal* if $(r_1+c_1)-(r_2+c_2)$ is divisible by n. An example of a secondary extended diagonal is $\{(1,1),(2,4),(3,3),(4,2)\}$.

Latin square is called *avangard* if each of its primary extended diagonals contains each number from 1 to n exactly once and each of its secondary extended diagonals also contains each number from 1 to n exactly once.

For example, the square shown above is not avangard, because its main diagonal $\{(1,1),(2,2),(3,3),(4,4)\}$ contains two instances of 2. There is actually no avangard Latin square of order 4.

Given n find avangard Latin square of order n.

Input

Input file contains one integer n ranging from 1 to 1000.

Output

The first line of the output file must contain "`Yes`" if there is an avangard Latin square of order n, or "`No`" if there is none.

If such square exists, the following n lines must contain n numbers each — the corresponding square. If there are multiple possible solutions, you can print any one.

Examples

avangard.in	avangard.out

`1`	`Yes`
	`1`

`2`	`No`

Problem B. Bicoloring of a Tree

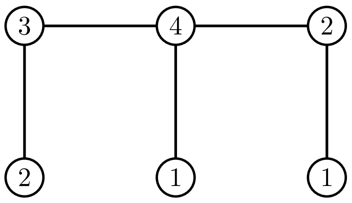
Input file:	`bicoloring.in`
Output file:	`bicoloring.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Connected undirected graph without cycles is called a tree. *Proper coloring* of a graph is a mapping of vertices of the graph to a set of *colors* so that no edge has its endpoints mapped to the same color.

Proper coloring of a graph is called *bicoloring* if:

- any color is used for at most two vertices;
- no two edges have the same pair of colors on their endpoints.

For example, the picture below show bicoloring of a tree. Colors are denoted by numbers from 1 to 4.



In this problem you must find bicoloring of a given tree with $2n$ vertices with at most $n + 1$ colors. It is known that such coloring exists if no vertex has more than n adjacent vertices.

Input

The first line of the input file contains $2n$ — the number of vertices of the tree ($2 \le 2n \le 100\,000$). The following $2n - 1$ lines contain tree edges. Each edge is specified by numbers of vertices that it connects. Vertices are numbered from 1 to $2n$. No vertex has more than n adjacent vertices.

Output

Output $2n$ numbers ranging from 1 to $n+1$. The i-th of the printed numbers must be the color of the i-th vertex. Each color must be used at most twice. Each pair of colors can correspond to colors of endpoints of at most one edge. If there are several possible solutions, you can print any one.

Examples

bicoloring.in	bicoloring.out

6
1 2
3 2
4 3
5 3
5 6
	2 3 4 1 2 1

Problem C. Commuter Trains

Input file:	`commuter.in`
Output file:	`commuter.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Railroads for commuter trains are often constructed with just one track. Trains moving in opposite direction pass each other at stations that have multiple tracks.

One commuter railroad has n stations arranged in a line. The i-th of the stations is located at a distance of d_i km from the first station. Each pair of adjacent stations is connected by a one-track railroad segment, so at any moment only trains moving in one direction can coexist at each of these segments.

Railroad administrators are planning to create a new time table. For each train they know its departure station, its destination station and departure time. We will consider all trains moving between stations at a constant speed of 1 km/min. Acceleration and deceleration time can be ignored. Each train must stop at any station it passes for at least 1 min.

To organize traffic the following rules are used. For each track segment the *waiting list* of trains that must move along it is created. After a train stops at a station other than its destination it stays there for one minute and then it is added to the waiting list that corresponds to the segment that it must enter. If the train is at its departure station it is added to the corresponding list at its departure time.

Each minute for each track that has non-empty waiting list the following operations are performed. A train that was put to the list earliest is selected. If there are several such trains, the train is selected that must move in the same direction that the trains that already exist at the track segment. If there are no trains at the segment, or there are several trains that must move in the same direction, the one is selected that has earlier departure time from its departure station. If there are still several such trains, the one that must move in direction from the station with greater number to the station with smaller number is selected.

After the train is selected, if there are no trains at the track segment, or the existing trains move in the same direction as the selected train, the selected train is removed from the waiting list and starts moving along the segment. In the other case, no train starts moving along this segment during this minute.

After the train arrives at some station different from its destination, it stops there for one minute. After that it is added to the waiting list of the track segment it must start moving along. Arriving of the train to the station as well as putting it to the list occur immediately before considering waiting lists of all segments.

You can assume that the number of tracks at any station is enough to store any number of trains.

Given departure stations, destination stations and departure times of all trains, create time table of the railroad. It is known that the total number of stations that all trains stop at doesn't exceed 300 000.

Input

The first line of the input file contains integer n ($2 \le n \le 100\,000$) — number of stations. The second line contains n integers: $d_1, d_2, \ldots, d_n$ ($0 = d_1 < d_2 < \ldots < d_n \le 10^9$). The third line contains m ($1 \le m \le 100\,000$) —– number of trains. It is followed by m lines, each of which describes one train. The description of the j-th train consists of three integers: s_j, f_j and t_j ($1 \le s_j, f_j \le n$, $s_j \ne f_j$, $0 \le t_j \le 10^9$) — its departure station, its destination station and its departure time in minutes. It is guaranteed that for any departure station at most one train is scheduled to depart at any given moment.

Output

For each train output the header "`Train` j", where j is the number of the train in the input file. After that print k lines, where k is the number of stations where the train stops. For each stop print its number followed by train arrival time to that station, followed by the train departure time from that station. Print asterisk "*" for arrival time to the departure station, similarly print asterisk for departure time from the last station. The total number of stations that all trains stop at doesn't exceed 300 000.

Examples

commuter.in	commuter.out
4	Train 1
0 3 5 10	1 * 0
4	2 3 4
1 4 0	3 6 7
4 1 0	4 12 *
1 3 10	Train 2
2 1 10	4 * 0
	3 5 6
	2 8 9
	1 12 *
	Train 3
	1 * 13
	2 16 17
	3 19 *
	Train 4
	2 * 10
	1 13 *

Problem D. Data Mining

Input file:	`data.in`
Output file:	`data.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Sometimes when considering sequences of various objects the values of objects are not important, but only relation which objects are equal to other objects. In such case canonization of the sequence can be considered.

Let us call two sequences of integers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ equivalent if there is a one-to-one mapping $\varphi : \mathbb{Z} \to \mathbb{Z}$ such that $b_i = \varphi(a_i)$ for all i. Lexicographically smallest sequence equivalent to the given one is called its *canonization*. For example, canonization of a sequence $3, 1, 4, 1, 5$ is $1, 2, 3, 2, 4$.

In various applications *suffix dictionaries* are used. Suffix of a sequence $a_1, a_2, \ldots, a_n$ is the sequence $a_i, a_{i+1}, \ldots, a_n$ for some i from 1 to n. Unfortunately canonization of suffix can be different from suffix of canonization. For example, in the sequence above the canonization of its suffix $4, 1, 5$ is $1, 2, 3$ which is different from $3, 2, 4$ — the corresponding suffix of its canonization.

In this problem you are given a sequence $a_1, a_2, \ldots, a_n$ and have to answer a series of queries of the following form: given i and p find the p-th number in the canonization of the suffix $a_i, a_{i+1}, \ldots, a_n$ of the given sequence.

Input

The first line of the input file contains n — the length of the given sequence ($1 \le n \le 200\,000$). The second line contains n integers: $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

The third line of the input file contains m — number of queries ($1 \le m \le 200\,000$). The following m lines contain two integers each, for each query its parameters i and p are given ($1 \le i \le n$, $1 \le p \le n - i + 1$).

Output

For each query output the value of the p-th element of canonization of the i-th suffix of the given sequence.

Examples

data.in	data.out

7
1 2 1 3 1 2 1
10
1 4
2 3
3 2
4 1
1 6
2 5
3 4
4 3
5 2
6 1
	3
	3
	2
	1
	2
	1
	3
	3
	2
	1

Problem E. Eating Chocolate Game

Input file:	`eating.in`
Output file:	`eating.out`
Time limit:	1 second
Memory limit:	256 megabytes

Infinite number of players are playing *Eating Chocolate Game*. Players are numbered by integers from 0 to infinity. Initially the players have three pieces of chocolate containing a_1, b_1 and c_1 chocolate squares, respectively, before them.

Players make their moves in turn, starting from player 1. The move of the i-th player consists of the following steps. Let the player have pieces of a_i, b_i and c_i squares before him. First he chooses on of the pieces and *secures* it. After that player $i - 1$ takes one of the unsecured pieces and eats it. After that player i divides one of the two remaining pieces into two parts of x and y squares, respectively. Both parts must be non-empty. If he cannot do it because both parts contain only one square, the game is over, he and all following players don't eat any chocolate. After that the move passes to player $i + 1$.

All players are greedy, cautious and minimalistic. That means that among all possible actions the player selects the one which allows him to eat as much chocolate as possible when he acts during the next player's move, regardless of other players actions (in particular, he doesn't assume other players play optimally). If there are several optimal possibilities the player chooses the one with minimal parameter. Namely, when selecting which piece to secure the player selects smallest piece among those letting him eat as much chocolate as possible. Similarly, when dividing a piece, the player chooses the optimal way to do it which has smallest possible $\min(x, y)$.

Help the players understand how much chocolate would each of them eat.

Input

The input file contains several test cases. Each test case consists of a line with three integers: a_1, b_1 and c_1 ($1 \le a_1, b_1, c_1 \le 10^{18}$). Input is followed by a line containing three zeroes, it must not be processed.

Output

For each test case print three lines. The first line must contain case number as shown in sample output. The second line must contain k — the number of players that would eat some chocolate. The third line must have k integers: for each player from 0 to $k - 1$ print the number of squares that he would eat.

Examples

eating.in	eating.out

8 2 1
1 1 1
0 0 0
	Case #1:
	4
	2 4 2 1
	Case #2:
	1
	1

In the first example the game proceeds as shown in the following table.

Player	Available pieces	Secured piece	Eaten piece	Division
1	8, 2, 1	8	2	$8 \to 4, 4$
2	4, 4, 1	1	4	$4 \to 2, 2$
3	2, 2, 1	1	2	$2 \to 1, 1$
4	1, 1, 1	1	1	game over

Problem F. Furniture Factory

Input file:	`furniture.in`
Output file:	`furniture.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

John is the director of a furniture factory. His factory is producing furniture in flat packages to be assembled by the client. This allows to produce different components of the same item in any order. However, the furniture that the factory is producing is luxurious and therefore there are some limitations for the process. Luxurious furniture is produced from rare valuable wood, so exact match of color and texture is required. Therefore it is impossible to produce different components simultaneously because all previous components are needed to be consistently checked and matched against the component being produced.

Recently John has accepted n orders for redwood escritoires. Each escritoire is constructed out of m components. Each component is produced by a special designated group of factory workers. Since escritoires are hand-crafted, it takes exactly one day to produce a component. Components of each escritoire can be produced in any order, but it is impossible to construct several components of the same escritoire on the same day. Each particular component can be produced for at most one escritoire each day.

For each escritoire John knows its due date d_i. If the escritoire is not ready by that date, John has to pay a penalty of v dollars to the corresponding client.

Help John to arrange production of escritoires so that his penalty was as small as possible.

Input

The first line of the input file contains three integers: n, m and v ($1 \le n \le 200$, $1 \le m \le 100$, $1 \le v \le 10^6$). The second line contains n integers: $d_1, d_2, \ldots, d_n$ ($1 \le d_i \le 1000$).

Output

The first line must contain p — the minimal total penalty John has to pay. The following n lines must contain construction plans for each of the escritoires. Each line must contain m integers: the j-th number in the i-th of these lines must be the day when the j-th component of the i-th escritoire must be produced. Day number must not exceed 10^5.

If there are several possible solutions, you can output any one.

Examples

furniture.in	furniture.out

5 3 100
3 4 5 4 5
	100
	1 2 3
	2 3 4
	3 4 5
	4 1 2
	5 6 1

In the given example it is impossible to construct all escritoires in time. One of them is late, so a penalty of 100 dollars must be paid.

Problem G. Genome Research

Input file:	`genome.in`
Output file:	`genome.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Scientists of Retrozavodsk Laboratory of Genome Research are preparing to analyze species from Pandora — a new 3D planet discovered at the nearby star. They have decided to start by analyzing genotypes of small bacteria called *E-lectus*.

After decoding its genome, the scientists have learned that there are m different genotypes of E-lectus. E-lectus lives in colonies and bacteria of various genotype are distributed in some proportion in each colony. This proportion is approximitely the same in each colony. One of genotypes, known as *unific* is dominating in each colony having more species than any other genotype.

Scientists decided to take several samples from each of the n colonies. Their funds are limited, so they hired a chinese company *Tzen Triz Bir Co* headed by Prof Chu Ro. The company took samples from each colony by sequencing genome of 50 to 1000 bacteria from it. However, Chu Ro has his own interest in analyzing Pandora nature, so he decided to spoil Retrozavodsk Lab study. So he ordered to make frauds of two types to the samples.

First they have chosen some colonies and added some bacteria with unific genotype to samples from these colonies.

Secondly they have chosen some other colonies and replaced some bacteria from these colonies that have genotype different from unific to species with unific genotype. Only bacteria with genotype that have at most 15% species in a total population could be replaced.

The resulting distorted distributions of genotypes among colonies were presented to Retrozavodsk Scientists. Fortunately, scientists are not idiots, so they understood that something is wrong with the data. Now they are trying to recover, data from which colonies were not frauded (type 0), data from which colonies were changed by adding some unific bacteria (type 1) and data from which colonies were changed by replacing some bacteria with unific bacteria (type 2).

You are given distributions as input. For each colony you have to detect its type.

Note

In this problem test generation and evaluation are different from traditional.

Each test case was generated using the following algorithm. First, integer n was selected between 200 and 1000 inclusive, and integer m is selected between 5 and 10 inclusive, index u of unific genotype is selected between 1 and m, inclusive. Distribution of genotypes $P = (p_1, p_2, \ldots, p_m)$ is selected ($0.01 \le p_i \le 0.50$, $p_1 + p_2 + \ldots + p_m = 1$, $p_u > p_i$ for all $i \ne u$). There is at least one other genotype which has $p_i > 0.15$ and at least two genotypes which have $p_i < 0.15$, sum of probabilities of all genotypes that have $p_i < 0.15$ is at least 0.20.

For each of n colonies number of sampled species k_i is selected uniformly at random between 50 and 1000. After that k_i times a species genotype is generated with distribution P. Denote number of species with corresponding genotypes as $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$.

Next two probabilities r_1 and r_2 are selected between 0.25 and 0.35 each. All n colonies are considered one after another. Each colony is left normal with probability $1 - r_1 - r_2$, is frauded by adding unific species with probability r_1 and is frauded by replacing some species by unific ones with proabability r_2.

Fraud by adding unific species is performed in the following way. Let k_i samples be taken from the corresponding colony. An integer f_i is generated uniformly at random between $0.25k_i$ and $1.25k_i$, inclusive.

Then f_i unific genotypes are added to the data ($a_{i,u} += f_i$).

Fraud by replacing some species with unific is performed in the following way. For each genotype such that $p_j \le 0.15$ an integer $f_{i,j}$ is selected between $0.75a_{i,j}$ and $a_{i,j}$, inclusive. Then $f_{i,j}$ species with genotype j are replaced by species with unific genotype ($a_{i,j} -= f_{i,j}$, $a_{i,u} += f_{i,j}$).

After that percentages of bacteria of corresponding genotype in each colony are rounded to exactly 2 digits after the decimal point. Name these percentages as $b_{i,j}$. They are provided in the input file.

You output will be accepted if you correctly identify at least 2/3 of colonies of each of types 0, 1 and 2.

Sample input has $n = 45$ to fit the page. Any correcly formatted output for sample input (test 1) will be accepted. It was generated using rules above with $n = 45$, $m = 7$, $P = (0.15, 0.10, 0.02, 0.25, 0.10, 0.35, 0.03)$, $r_1 = r_2 = 0.30$.

Judges' program for general case correctly solves samples input without any additional suggestions, making only two incorrect identifications.

Input

The first line of the input file contains n and m ($200 \le n \le 1000$, $5 \le m \le 10$, except sample input which has $n = 45$).

The following n lines contain m real numbers given with exactly two digits after the decimal point — $b_{i,j}$. Due to rounding errors they may not sum exactly to 100.

Output

For each colony output its type — 0, 1 or 2.

Examples

genome.in	genome.out
45 7	
10.67 7.11 0.91 16.71 6.04 57.24 1.32	1
10.08 6.79 1.71 16.26 5.90 56.86 2.40	1
3.70 2.47 1.23 25.93 2.47 64.20 0.00	2
14.31 9.19 1.94 28.27 8.13 36.04 2.12	0
3.23 1.08 1.08 29.03 2.15 63.44 0.00	2
2.36 0.79 0.52 25.72 0.52 69.82 0.26	2
13.75 9.82 1.96 24.75 11.98 34.18 3.54	0
1.19 0.00 0.40 21.43 1.98 74.21 0.79	2
14.78 7.26 2.69 26.61 10.75 35.22 2.69	0
0.54 0.98 0.22 24.21 0.11 73.62 0.33	2
17.20 9.47 1.87 21.73 9.87 36.67 3.20	0
10.06 12.43 2.37 23.67 12.43 37.87 1.18	0
13.78 9.34 1.68 25.11 9.80 37.37 2.91	0
0.85 0.71 0.28 24.96 0.56 72.50 0.14	2
18.02 8.71 2.10 26.43 9.91 32.43 2.40	0
15.96 9.64 2.01 22.69 9.24 36.95 3.51	0
3.07 2.50 0.58 25.72 0.77 67.18 0.19	2
1.33 0.88 0.11 24.34 2.21 71.02 0.11	2
16.25 9.05 2.84 25.63 9.71 33.15 3.38	0
12.70 7.94 0.79 24.60 15.08 32.54 6.35	0
14.73 9.45 0.18 25.45 7.82 37.82 4.55	0
0.35 1.39 0.69 25.00 0.35 72.22 0.00	2
8.02 5.73 1.53 17.18 6.49 60.31 0.76	1
13.99 7.77 2.07 23.83 11.40 37.65 3.28	0
3.05 0.55 0.14 26.87 0.83 68.01 0.55	2
13.35 10.96 0.80 26.69 12.15 33.86 2.19	0
13.33 15.24 2.54 21.27 9.21 34.92 3.49	0
7.63 5.10 0.82 12.46 5.21 67.51 1.26	1
16.42 10.22 1.82 22.63 12.04 33.39 3.47	0
3.51 1.00 0.75 27.32 0.75 66.67 0.00	2
14.29 9.52 1.86 25.05 10.35 35.20 3.73	0
11.65 4.82 2.01 18.88 7.63 52.61 2.41	1
13.16 11.28 3.01 24.44 12.03 32.33 3.76	0
0.13 0.78 0.39 23.28 1.17 73.99 0.26	2
0.51 1.03 1.03 27.18 1.03 68.72 0.51	2
14.89 9.24 1.52 25.22 10.87 33.70 4.57	0
17.46 9.66 3.22 23.90 8.98 34.75 2.03	0
13.91 13.53 1.88 24.25 8.08 34.77 3.57	0
14.40 9.89 1.65 25.16 9.56 36.37 2.97	0
15.88 12.42 2.57 25.28 10.40 30.87 2.57	0
1.94 1.77 0.00 28.87 0.97 66.29 0.16	2
14.07 8.89 2.22 22.22 17.04 31.11 4.44	0
6.45 3.90 0.78 12.48 4.26 70.78 1.35	1
15.13 8.79 1.02 23.31 10.43 37.42 3.89	0
15.21 10.84 2.27 21.68 11.49 34.63 3.88	0

Problem H. House of Representatives

Input file:	`house.in`
Output file:	`house.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

After revolution in Flatland its parliament building was destroyed by fire, so it was decided to build new House of Representatives. Since revolutionaries decided that there will be no capital in new free Flatland, it is not easy to choose the city where the House must be built.

Flatland has n cities. Flatland has neither oil, nor innovative modernized production, so its budget is very limited. Therefore there are only $n-1$ roads in Flatland, each road is bidirectional and connects two cities. For each road its length is known. Fortunately, it is possible to get from any city to any other one by roads. The population of the i-th city is p_i thousand citizens.

After some discussion the members of the Revolution Committee that are now governing the country decided that the new House of Representatives must be as accessible as possible. For each city u its inaccessiblity is calculated using the following formula: $f(u) = \sum_{v} \rho(v,u)p_v$. Here $\rho(u,v)$ is the distance between cities u and v.

Help the Committee to choose the location for the House so that its inaccessability was as small as possible.

Input

The first line of the input file contains n ($1 \le n \le 100\,000$). The second line contains n integers: $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 100\,000$). The following $n-1$ lines describe roads, the i-th road is described by three integers: a_i, b_i and l_i — numbers of cities it connects, and its length ($1 \le l_i \le 1000$).

Output

Output two integers: u and $f(u)$, u must be the number of the city where the House of Representatives must be built. If there are several optimal locations, you can output any one.

Examples

house.in	house.out

6
6 3 5 4 2 1
1 3 3
2 3 5
3 4 7
4 5 10
4 6 2
	3 104

Problem I. Immetric Polynomials

Input file:	`immetric.in`
Output file:	`immetric.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Professor Nalatac calls a polynomial $p(x, y)$ in two variables *immetric* if all of its coefficients are 0 and 1, and $p(x, 1 - x)$ is equal to 1 for all real x. Recall that degree of a polynomial p is maximal $k + m$ such that p contains monomial $x^k y^m$.

For example, polynomials $x + xy + y^2$ and $x^2 + xy + y$ are both immetric. These polynomials are the only two immetric polynomials of degree 2.

Now he is interested in counting immetric polynomials of a given degree. Help him with this problem.

Input

Input file contains several test cases. Each test case consists of a number n on a line by itself ($1 \le n \le 9$). The last test case is followed by a line with $n = 0$, it should not be processed.

Output

For each test case print the number of immetric polynomials of degree n.

Examples

immetric.in	immetric.out
1 2 3 4 0	Case #1: 1 Case #2: 2 Case #3: 5 Case #4: 14

Problem J. Jealous Robots

Input file:	`jealous.in`
Output file:	`jealous.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Robot I215 is very jealous. Recently he discovered that his robogirlfriend K776 played 5-dimensional chess with his rival robot U666, so he decided to call him to a duel.

Duel between robots will use the following rules. The duel will take place on a rectangular grid of $m \times n$ unit squares. Some squares are empty, others are completely occupied by impassable walls. First U666 chooses one empty location and stands in the center of the corresponding cell. After that I215 chooses another empty location and also stands in the corresponding center. Finally U666 can now move to any empty cell that is either in the same row, or in the same column as his initial location and all cells between his initial final location and his final location are also empty. He can also stay in his location.

If after his turn I215 can see U666, he shoots at him with laser and kills him. Robots are quite small, so we will consider them being points on the plane. A point A is visible from point B if the segment connecting them neither intersects nor touches any wall.

Robot U666 thinks that he is not guilty. Actually he just loves 5D chess very much. Help him find such cell in a grid that regardless of I215's action he could move to a safe place. Let us call such cells *good*. In fact he is interested in finding all good cells.

Input

The first line of the input file contains two integers: m and n ($1 \le m, n \le 30$, $mn \ge 2$. The following m lines contain n characters each. The j-th character of the i-th line is "`.`" if the corresponding cell is empty, or "`#`" if it is occupied by a wall. All empty cells form a connected set (it is possible to get from any cell to any other by moving from a cell to adjacent cell, two cells are adjacent if they share a side). There are at least two empty cells.

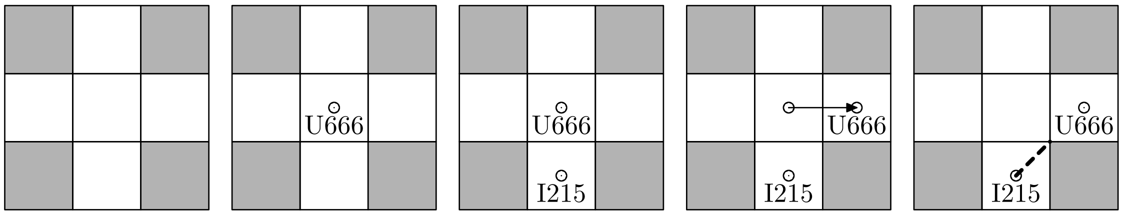
Output

The first line of the output file must contain k — the number of good cells. The following k lines must list all good cells. Each cell must be specified by its row followed by a space followed by its column. Rows are numbered from 1 to m starting from the top. Columns are numbered from 1 to n, starting from the left. Print good cells in non-decreasing order of their row, cells with equal row in increasing order of their column.

Examples

jealous.in	jealous.out

`3 3`
`#.#`
`...`
`#.#`
	`1`
	`2 2`


U666
U666
I215
U666
I215
U666
I215