Problem E. Quantization Problem

Input file:	`quant.in`
Output file:	`quant.out`
Time limit:	0.5 seconds

When entering some analog data into a computer, this information must be quantized. Quantization transforms each measured value x to some other value $l(x)$ selected from the predefined set L of levels. Sometimes to reduce the influence of the levels set to the information, the group of levels sets L_i is used. The number of levels sets is usually chosen to be the power of 2.

When using the number of levels sets, some additional information should be used to specify which set was used for each quantization. However, providing this information may be too expensive — the better solution would be to choose more levels and use one set. To avoid the specification of the quantization set, the following technique is used. Suppose that n values $x_1, x_2, \ldots, x_n$ are to be quantized and the group of $m = 2^p$ levels sets $\{L_i\}_{i=0}^{m-1}$ each of size $s = 2^q$ is used to quantize it. After quantization x_j is replaced with some number $l_j =\in L_{f(j)}$. Instead of sending l_j, its ordinal number in $L_{f(j)}$ is usually sent, let k_j be the ordinal number of l_j in $L_{f(j)}$ (levels are numbered starting with 0). Take p least significant bits of k_j and say that the number $k_j \& (2^p - 1)$ is the number of the levels set that will be used for next quantization, that is $f(j+1) = k_j \& (2^p - 1)$.

Since the least significant bits of k_j are usually distributed quite randomly, the sets used for quantization change often and weakly depend on values of quantized data, thus this technique provides the good way to perform the quantization.

Usually to perform the quantization the closest to the value level of the levels set is chosen. However, using the technique described, sometimes it pays off to choose not the optimal level, but some other one, the ordinal number of which has other least significant bits, thus choosing another levels set for next measure and providing better approximation of quantized values in the future. Let us call the *deviation* of quantization the value $\sum_{j=1}^{n} |x_j - l_j|$. Your task is given measures and levels sets to choose quantized value for each measure in such a way, that the deviation of quantization is minimal possible.

The first value is always quantized using set L_0.

Input

The first line of the input file contains n ($1 \le n \le 1000$). The second line contains n integer numbers x_i ranging from 1 to 10^6. The next line contains m and s ($1 \le m \le 128$, $m \le s \le 128$). Next m lines contain s integer numbers each — levels of the quantization sets given in increasing order for each set, all levels satisfy $1 \le level \le 10^6$.

Output

First output the minimal possible deviation of the quantization. Then output n integer numbers in range from 0 to $s - 1$. For each input value output the number of the level in the corresponding levels set (k_j) used for this number to achieve the quantization required.

Example

quant.in	quant.out
3 8 8 19 2 4 5 10 15 20 3 7 13 17	5 1 1 3

Problem A. Little Brackets

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

Consider all regular bracket sequences with one type of brackets. Let us call the *depth* of the sequence the maximal difference between the number of opening and the number of closing brackets in a sequence prefix. For example, the depth of the sequence "`()()(())`" is 2, and the depth of "`(((()(()))))`" is 4.

Find out the number of regular bracket sequences with n opening brackets that have the depth equal to k. For example, for $n = 3$ and $k = 2$ there are three such sequences: "`()(())`", "`(()())`", "`(())()`".

Input

Input file contains several test cases. Each test case is described with n and k ($1 \le k \le n \le 50$).

Last testcase is followed by two zeroes. They should not be processed.

Output

For each testcase output the number of regular bracket sequences with n opening brackets that have the depth equal to k.

Separate output for different testcases by a blank line. Adhere to the format of the sample output.

Example

brackets.in	brackets.out
3 2 37 23 0 0	Case 1: 3 Case 2: 203685956218528

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 A. The Smart Bomb

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

The military scientists of Flatland have recently developed the new smart bomb. The main feature of the bomb is that its explosion power can be regulated after the production.

After the bomb was added to the arsenal, the generals have decided to run a series of test explosions. The equipment department has provided them with three bombs to explode. Since bureaucracy is a significant problem in the modern Flatland, the generals have decided to make all three explosions in one day. They selected the points where the bombs must be placed, and started to choose the destructing power of the bombs to be exploded.

Of course, the generals want to explode as powerful bombs as possible. On the other hand, to analyze the results of the experiment, the craters of the explosions must not intersect. Each crater is a circle, its radius is proportional to the power of the bomb exploded.

Now the generals want to know what is the maximal total power of the bombs they can explode. Help them to find that out! The total power of the bombs is the sum of their powers.

Input

The input file contains three lines, each contains two integer numbers — the coordinates of the location where the corresponding bomb must be exploded. Coordinates do not exceed 10^9 by their absolute value. Points do not belong to the same line.

Output

Output three real numbers — for each bomb output the desired radius of the crater after its explosion. The sum of the radii must be maximal possible, but craters must not intersect (although they may touch each other). Your answer must be accurate up to 10^{-4}.

Example

bomb.in	bomb.out
0 0 4 0 4 3	3.000000000 1.000000000 2.000000000

Problem A. Beer Problem

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

Everyone knows that World Finals of ACM ICPC 2004 were held in Prague. Besides its greatest architecture and culture, Prague is world famous for its beer. Though drinking too much is probably not good for contestants, many teams took advantage of tasting greatest beer for really low prices.

A new beer producing company *Drink Anywhere* is planning to distribute its product in several of the n European cities. The brewery is located near Prague, that we would certainly call city number 1. For delivering beer to other cities, the company is planning to use logistics company *Drive Anywhere* that provides m routes for carrying goods. Each route is described by the cities it connects (products can be transported in either direction), its capacity — the number of barrels of beer that can be transported along this route each day, and the cost of transporting one barrel of beer along it. To deliver beer to some city it may be necessary (or advantageous) to use several routes consequently, and maybe even deliver beer using several different paths.

Each city is in turn characterized by the price that local pubs and restaurants are ready to pay for one barrel of beer. You may assume that demand for beer is essentially unlimited in each city, since this is the product that will always find its consumer.

Drink Anywhere is not planning to distribute its beer in Prague for a while, because of the high competition there, so it is just planning to provide beer to other cities for now. Help it to find out, what is the maximal income per day it can get.

Input

The first line of the input file contains n and m — the number of cities in Europe we consider and the number of delivery routes respectively ($2 \le n \le 100$, $1 \le m \le 2000$). The next line contains $n-1$ integer numbers — prices of a barrel of beer in European cities $2, 3 \ldots, n$ respectively (prices are positive integers and do not exceed 1000).

The following m lines contain four integer numbers each and describe delivery routes. Each route is specified by the numbers of cities it connects, its capacity, and the price of transporting one barrel of beer along it (the capacity and the price are positive integers, they do not exceed 1000).

Output

Output the maximal income the company can get each day.

Example

beer.in	beer.out
4 4 80 50 130 1 2 80 50 2 4 40 90 3 1 40 60 3 4 30 50	3000

The company should deliver 80 barrels of beer to the second city (using the first route it costs 50 per barrel to deliver beer, income is 30 per barrel, 2400 total), and 30 barrels to the fourth city (the best path uses routes 3 and 4, it costs 110 to deliver a barrel, income is 20 per barrel, 600 total). It is not profitable to deliver beer to the third city, so the company should not do it.

Problem D. Integer Numbers

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

The boy likes numbers. He has a sheet of paper. He have written a sequence of consecutive integer numbers on the sheet. The boy likes them.

But then the girl came. The girl is cruel. She changed some of the numbers.

The boy is disappointed. He cries. He does not like all these random numbers. He likes consecutive numbers. He really likes them. But his numbers are not consecutive any more. The boy is disappointed. He cries.

Help the boy. He can change some numbers. He would not like to change many of them. He would like to change as few as possible. He cannot change their order. He would like the numbers to be consecutive again. Help the boy.

Input

The first line of the input file contains n — the number of numbers in the sequence ($1 \le n \le 50\,000$). The next line contains the sequence itself — integer numbers not exceeding 10^9 by their absolute values.

Output

Output the minimal number of numbers that the boy must change. After that output the sequence after the change.

Example

integer.in	integer.out
6 5 4 5 2 1 8	3 3 4 5 6 7 8

Problem C. Express Trains

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

The ministry of transport of Flatland has decided to build the new system of experss railways for the special extremely fast trains. But the budget of the country is limited because of the high course of the fleugric, the Flatland national currency. Therefore the ministry was forced to put some strong restrictions on the system of the railways.

After a discussion it was decided that the railways must be organized in such a way that there were no need for the complicated control system. That is, the railways must be running from west to east, without forks. Each railway must connect two cities, and may pass through some intermediate cities. Since the citizens of the country are concerned about its ecology, there must be at most one railway passing through each city, including the cities at the ends of the railway. Since there is not enough money to make a special infrastructure for the railways, the railways must go along existing roads.

After further inspection it was found out that the funds allocated for the project are only enough to build two railways. The minister has decided that the most important task at the moment is to connect the capital of the country A to the city B where the main Flatland university is located, and the city C where the big machine plant is operating to the main country port D. Thus, the two railways must run from west to east and connect A with B, and C with D, respectively.

Help the minister to find the way to build the railways.

Input

The first line of the input file contains n and m — the number of the cities in Flatland, and the number of existing roads ($4 \le n \le 2000$, $0 \le m \le 100\,000$). The second line contains four different numbers — the numbers of cities A, B, C and D, respectively.

Let cities be numbered from west to east. No two cities in Flatland are on the same longitude. The following m lines describe roads. Each line contains two numbers — a_i and b_i — the cities connected by the corresponding road ($a_i < b_i$). There can be several roads between a pair of cities.

Flatland is quite small, so there is no cycle of roads that goes around the planet.

Output

On the first line of the output file print "`YES`", if it is possible to build two railways that satisfy all the conditions described, and "`NO`" in the other case.

If it is possible to build the railways, the following two lines must contain the descriptions of the railways. Each description must be a sequence of the cities the railway goes through, from west to east.

Example

express.in	express.out
6 8 1 6 5 2 1 2 2 3 1 3 2 4 4 5 4 6 3 6 3 4	YES 1 3 6 2 4 5
5 4 1 2 4 5 1 3 2 3 3 4 3 5	NO

Problem F. Nonequal Parts

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

Alice and Bob play the following game. They put a heap of n coins to the table, and after that make moves in turn. Alice makes her move first.

Each move the player takes any heap of coins from the table, and splits it into several unequal parts. For example, a heap of 7 coins can be split in the following ways:

$$7 \to 6, 1$$
$$7 \to 5, 2$$
$$7 \to 4, 3$$
$$7 \to 4, 2, 1$$

The resulting heaps are put back to the table and the game continues. The player who has no valid moves loses.

Help Alice to find out whether she can win regardless of Bob's moves, and if she can what is the first move she must make.

For example, if $n = 7$, Alice can win by splitting the initial heap into heaps containing 4, 2 and 1 coins. Bob is forced to split the heap of 4 coins into two, containing 3 and 1 coins respectively, and Alice wins by splitting 3 to 2 and 1. There are now two heaps with 2 coins and 3 heaps with 1 coin, and none of them can be split further. So Bob loses.

Input

Input file contains one number n ($3 \le n \le 300$).

Output

If Alice can win regardless of Bob's moves, print "`win`" on the first line of the output file. On the second line print the sizes of heaps she should split the initial heap to in order to win.

If Bob can force Alice to lose, print "`lose`" on the first line of the output file.

Example

nonequal.in	nonequal.out
7	win 1 2 4
8	lose

Problem G. Primitive Product

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

The number $x \in \mathbb{Z}$ is called a *unit* if there exists $y \in \mathbb{Z}$ such that $xy = 1$. The number $x \in \mathbb{Z}$ is called *irreducible* if whenever $x = yz$, either y, or z is a unit.

Consider $n \in \mathbb{Z}$. We call its representation as a product

$$n = a_1 \cdot a_2 \cdot \ldots \cdot a_k$$

primitive if each a_i is irreducible and is not unit.

Given n, you have to find all ways to represent it as a primitive product. Two representations that only differ by the order of the factors are considered the same. The factors in each product must be sorted in the nondecreasing order. The representations must be sorted by the first factor, then by the second factor, and so on.

Input

Input file contains n ($2 \le |n| \le 2 \cdot 10^9$).

Output

On the first line of the output file print t — the number of representations of n as a primitive product. The following t lines must contain representations, one on a line. Adhere to the format of sample output.

Example

product.in	product.out
30	4 -5 -3 2 -5 -2 3 -3 -2 5 2 3 5

Problem J. Trip Expenses

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

In Janap business trip expenses compensation is not based on the travel documents that the person presents to the accountant, but rather on the starting point of the trip and the destination of the trip. The means of transportation for the trip can be chosen by the one who is assigned to the trip on his own.

Of course, those who would like to save money choose the cheapest way of travelling between cities. However such way can often be composed of several segments and therefore take longer.

The Ministry of Finances of Janap is investigating the average number of trip segments required to get from one city to another. They suppose that the travelling person chooses the cheapest path, and among those he chooses the path with the smallest number of segments.

Help them to find that out.

Input

The first line of the input file contains two integer numbers n and m ($2 \le n \le 300$, $1 \le m \le 20\,000$) — the number of cities in Janap and the number of possible ways to get from one city to another — trip segments. The following m lines describe trip segments, each segment is described by three integer numbers a, b and c — the cities it connects and the cost of travelling along this segment. Each segment can be traveled in either direction. There can be several segments between two cities. The cost of travelling is positive and doesn't exceed 10^9. It is possible to get from any city to any other.

Output

Output one floating point number — the average number of segments required to get from one city to another by cheapest ways. You answer must be accurate up to 10^{-5}.

Example

trip.in	trip.out
3 3 1 2 10 2 3 2 1 3 3	1.3333333333333333

In the example the cheapest path from 1 to 2 has two segments ($1 \to 3 \to 2$), the cheapest paths from 1 to 3 and from 2 to 3 have one segment each. The average number of segments if $(2 + 1 + 1)/3 = 4/3$.

Problem H. Hard Test

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

Andrew is having a hard time preparing his 239-th contest for Petrozavodsk. This time the solution to the problem is based on Dijkstra algorithm and Andrew wants to prepare the hard test for the algorithm.

The Dijkstra algorithm is used to find the shortest path from a source vertex to all other vertices in a graph. The algorithm acts as follows. Let G be a weight directed graph with vertex set V, edge set E and weight function $w : E \to \mathbb{R}^+$. Let all vertices be reachable from vertex s. The algorithm uses a set of vertices U, first initialized as empty. Each vertex is labeled with either an integer number, or with $+\infty$. Initially all vertices are labeled with $+\infty$, and the vertex s is labeled with 0. Denote the label of vertex v as $d[v]$.

A step of the algorithm is the following: the vertex with the minimal label that doesn't belong to U is selected. Let this vertex be u. The vertex u is added to the set U, and each edge $uv \in E$ is *relaxed*. The relaxation replaces $d[v]$ with $\min(d[v], d[u] + w(uv))$. The algorithm is over when all vertices belong to U. If the label of the vertex v has changed, the relaxation is said to be *active*.

Now Andrew would like to create a graph with n vertices and m edges, such that the Dijkstra algorithm makes as many active relaxations as possible. Help him to create such graph. To avoid nondeterminism, each time when selecting a vertex with minimal label among vertices that are not in U there must be exactly one vertex with the minimal label.

Input

The first line of the input file contains two integer numbers: n and m — the number of vertices and the number of edges in the graph Andrew would like to create ($4 \le n \le 1000$, $n - 1 \le m \le n^2/5$).

Output

Output m lines — the edges of the graph. Each line must contain three integer numbers: the beginning of the edge, the end of the edge and the weight of the edge. All weights must be non-negative and must not exceed 10^6. All vertices must be reachable from vertex 1. If Dijkstra algorithm is run with $s = 1$ there must be maximal possible number of active relaxations among all graphs with n vertices and m edges. There must be no loops and no parallel edges.

Example

test.in	test.out
4 3	1 2 0 1 3 1 1 4 2
5 5	1 2 0 1 3 1 2 4 4 3 4 2 1 5 2

Problem D. Avoiding Partitions

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

A partition of a set $S = \{1, 2, \ldots, n\}$ is a collection $\mathcal{P}$ of sets $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ such that $\bigcup P_i = S$ and $P_i \cap P_j = \varnothing$ for $i \neq j$. An example of a partition for $n = 5$ is $P_1 = \{1, 3\}$, $P_2 = \{2, 4, 5\}$.

The partition is said to avoid set Q if none of P_i has Q as a subset. For example, the partition above avoids sets $\{1, 2\}$ and $\{3, 4\}$ but doesn't avoid $\{1, 3\}$ nor $\{2\}$.

Given n and a collection of sets $Q_1, Q_2, \ldots, Q_l$ find the number of partitions of S that avoid each of Q_i.

Input

The first line of the input file contains two integer numbers n and l ($1 \le n \le 100$, $0 \le l \le 10$). The following l lines describe sets to avoid. Each line starts with one integer number q_i — the size of the set, followed by q_i numbers — the elements of the set.

Output

Output one integer number — the number of partitions avoiding each of Q_i.

Example

partitions.in	partitions.out
5 2 3 1 2 3 2 2 4	34

Problem C. Spending Budget

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

As everyone knows the most important part of any governmental project is spending budget money. There are currently m national projects running in Flatland, the i-th project is capable of spending s_i billions dollar per day.

The government of Flatland is planning to allocate n money lots to finance the projects, each lot equal to p billion dollar. The money of the i-th lot would be available for spending in the beginning of the r_i-th day. After the money of a lot are available, they can be assigned to some project. This projects starts to *draw* this money lot and spends $\lceil p/s_i \rceil$ days doing so. While the project draws money of some lot it cannot be assigned another lot.

The Minister of Finances of Flatland wonders, how soon can all the money be spent. Help him to find out the minimal possible date that the money can be completely spent.

Input

The first line of the input file contains m, n and p ($1 \le m \le 100$, $1 \le n \le 100$, $1 \le p \le 10^9$). The second line contains m integer numbers: $s_1, s_2, \ldots, s_m$ ($1 \le s_i \le 10^9$). The third line contains n integer numbers: $r_1, r_2, \ldots, r_n$ ($1 \le r_i \le 10^9$).

Output

The first line of the output file must contain one integer number — the minimal possible day on which drawing money of all lots can be completed.

Examples

budget.in	budget.out
2 4 22 2 5 1 3 8 12	17

One optimal scheme of drawing money is the following: the lot received on day 1 is assigned to project 1 (it takes 11 days to complete it). The other lots are assigned to project 2, lot 2 takes days 3–7, lot 3 takes days 8–12, lot 4 takes days 13–17 (note that lot 4 is available on day 12, but is assigned only on day 13).