Problem A. Apple

Input file:	`apple.in`
Output file:	`apple.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

I'll call the problem "Apple" if by 5 o'clock you have not offered anything better

Problem developer, the night before the contest

Kostya had an apple. He divided it into n parts. Then he bit i-th part a_i times. Help him calculate

$$\frac{2^{n-1}}{2 \cdot \pi} \int_0^{2\pi} \cos(a_1 x) \cdot \cos(a_2 x) \cdots \cos(a_n x)\, dx.$$

Input

The first line of input contains an interger n ($1 \le n \le 100$). The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 1000$).

Output

Output the answer rounded to the nearest integer. It is guaranteed that the answer will not be within 10^{-5} of a number $c + 0.5$ for some integer c.

Example

apple.in	apple.out
2 1 1	1

Problem B. Clock

Input file:	`clock.in`
Output file:	`clock.out`
Time limit:	1 second
Memory limit:	256 mebibytes

A mechanical clock hangs on the wall. The clock has three hands: an hour hand which is at least 100 units long, a minute hand (twice as long as the hour hand) and a second hand (three times as long as the hour hand). As time passes, each hand of the clock rotates continuously with constant speed in the clockwise direction such that the hour hand makes a full turn in 12 hours, the minute hand in one hour and the second hand in one minute. The center of rotation is the same for all three hands. At 12:00:00, all three hands point strictly upwards.

You are given a rasterized black-and-white photo of this clock (or a part of it). All three hands of the clock are fully visible on the photo. One unit of length equals the width and height of one pixel on the rasterized photo. Each hand looks like a rasterized segment ≈ 1 pixel thick. However, due to rounding errors and sun fluctuations, it can appear thinner or thicker at any particular point.

More formally, rasterization is performed in the following way. First, for each hand, the coordinates of the segment representing that hand are rounded down to nearest integers. Next, all pixels with centers at distance ≤ 0.55 from this segment are painted black. After all hands have been rasterized, all pixels which are not painted black are considered white.

Your goal is to find the time shown by the clock. To simplify your task, the following is guaranteed:

- the number of seconds passed from the start of the current minute is an integer,
- the center of rotation of hands is located exactly at the center of some pixel of the photo,
- each two hands form an angle with a measure from 5 to 175 degrees,
- there is a white border at least 10 pixels thick, that is, black pixels are located inside $10 \leq x < w - 10$, $10 \leq y < h - 10$ where w and h are dimensions of the photo.

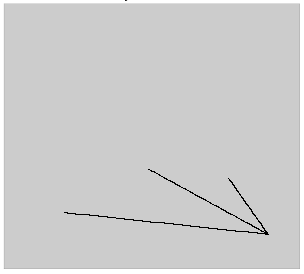
Input

First line contains two integers w and h: the dimensions of the photo ($1 \leq w, h \leq 2000$). Each of the next h lines contains w characters which are '`.`' for white pixels and '`#`' for black ones.

Output

Output the current time with leading zeroes. Hours are numbered 12, 01, 02, 03, ..., 11. Minutes and seconds are numbered from 00 to 59.

Example

clock.in	clock.out	Notes
`w = 625, h = 560` 	`10:49:46`	You can download the example input at `http://forest.acm.petrsu.ru/130130/clock.in` (use your team ID and password to get access to the file).

Problem C. Equality

Input file:	`equality.in`
Output file:	`equality.out`
Time limit:	4 seconds
Memory limit:	256 mebibytes

Once upon a time there was a biologist called Ksenia. She was interested in the life of microorganisms. One day she went to a shop and bought three bacteria. She took her microscope and started watching them. Soon Ksenia found that all three bacteria were of different types: red, green and blue. She also found simple laws for their reproduction. At the end of each day, each red bacterium divides into exactly x red bacteria, each green bacterium divides into exactly y green bacteria, and each blue bacterium divides into exactly z blue bacteria. Ksenia called these x, y, z the *parameters* of bacteria.

The biologist was watching them for five months, and at the end of each day, she wrote down the current amount of bacteria of each color. She had three colored notepads, one for each bacterium type, and she wrote each value on its own page. For example, on the first page of the red notepad, Ksenia wrote number x (amount of red bacteria at the end of the first day), on the second page of the green notepad, she wrote y^2 and so on. At last, Ksenia looked through her notes and found a very interesting equation: she took the number on page a of the red notepad, added it to the number on page b of the green notepad and found the result in the blue notepad on page c.

Years have passed since that day. Ksenia has forgotten the exact values of x, y and z and also the numbers of pages a, b and c which she used to obtain the equation. All she remembers is that $x_{\min} \le x \le x_{\max}$, $a_{\min} \le a \le a_{\max}$, $y_{\min} \le y \le y_{\max}$, $b_{\min} \le b \le b_{\max}$, $z_{\min} \le z \le z_{\max}$ and $c_{\min} \le c \le c_{\max}$. Now she is interested in a non-biological matter, which is, how many possible x, a, y, b, z and c, if any, satisfy the above conditions and the equation: the sum of the number on page a of the red notepad and the number on page b of the green notepad is equal to the number on page c of the blue notepad.

Can you help the biologist? Dealing with numbers is not exactly her profession, you know.

Input

The first line contains boundaries for x: $1 \le x_{\min} \le x_{\max} \le 150$.

The second line contains boundaries for a: $1 \le a_{\min} \le a_{\max} \le 150$.

The third line contains boundaries for y: $1 \le y_{\min} \le y_{\max} \le 150$.

The fourth line contains boundaries for b: $1 \le b_{\min} \le b_{\max} \le 150$.

The fifth line contains boundaries for z: $1 \le z_{\min} \le z_{\max} \le 150$.

The sixth line contains boundaries for c: $1 \le c_{\min} \le c_{\max} \le 150$.

Output

On the first line, print the amount of such numbers x, a, y, b, z and c.

Example

equality.in	equality.out
1 1 1 2 1 3 1 4 1 5 2 7	4

Explanation

The following solutions fall into example's boundaries: $1^1 + 2^3 = 3^2$, $1^1 + 3^1 = 2^2$, $1^2 + 2^3 = 3^2$ and $1^2 + 3^1 = 2^2$.

Problem D. Training

Input file:	`even.in`
Output file:	`even.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Vasya and Vasya are training hard for the tandem cycling marathon. They need to choose a route to train on.

Vasya and Vasya live in a small country with N cities and M bidirectional roads connecting them. A training route starts in some city, follows some roads and ends in the same city it started in. Vasya and Vasya like to see new places, so they made a rule not to visit any city twice and not to travel along the same road twice. The training route may start in any city and does not need to visit every city. Riding in the back seat is easier since the rider is shielded from the wind by the front rider. Because of this, Vasya and Vasya change seats in each city they arrive to. To ensure that they get the same amount of training, they must choose a route with an even number of roads.

Help Vasya and Vasya find such a training route.

Input

The input contains one or more test cases. The first line contains T, the number of test cases. The test cases follow.

The first line of each test case contains two integers N and M: the number of cities and the number of roads ($2 \le N \le 100\,000$, $1 \le M \le 300\,000$). Next M lines contain numbers of cities connected by roads. Cities are numbered from 1 to N. No road connects a city with itself, no two roads connect the same pair of cities.

It is guaranteed that the sum of N in all test cases does not exceed 100 000, and the sum of M in all test cases does not exceed 300 000.

Output

Output the answers for the test cases.

For each test case, on the first line, print the length of the training route you found, or `-1` if no route exists. If a route exists, print an additional line containing the numbers of cities in the route in the order of visiting.

If there are several possible solutions, output any one of them. Note that the length of the route can be any even positive integer.

Example

even.in	even.out
2 4 4 1 3 3 2 2 4 4 1 5 5 1 3 3 2 2 4 4 5 5 1	4 1 3 2 4 -1

Problem E. Factorial

Input file:	fact.in
Output file:	fact.out
Time limit:	1 second
Memory limit:	256 mebibytes

This part of problem statement is intentionally blank.

Input

The first line of input contains $1 \le n \le 2^{42}$.

Output

You should output $(2n-1)!! \bmod 2^{42}$.

Examples

fact.in	fact.out
3	15
239	149788644671

Explanation

In the first example, we have $(2 \cdot 3 - 1)!! = 5!! = 1 \cdot 3 \cdot 5 = 15$.

Problem F. Greedy

Input file:	greedy.in
Output file:	greedy.out
Time limit:	1 second
Memory limit:	256 mebibytes

In this problem, you have to solve a variation of the well-known knapsack problem. Unfortunately, this variation is not NP-hard.

You have n objects, each of them has size s_i and cost c_i. You also have q knapsacks, each of them is fully characterized by its size x_i. A knapsack of size x can contain some objects of total size at most x.

To fill a knapsack, you use the following greedy algorithm:

1. Sort all objects in non-increasing order by their sizes.
2. Iterate over the objects in that order.
3. For each object, if it can be put into the knapsack (free space in knapsack is enough), put it, otherwise skip it.

For each of the q knapsacks, run the described algorithm and then find the total cost of objects inside it.

Input

The first line of input contains one integer n: the number of objects ($1 \le n \le 10^5$).

The second line contains n integer sizes s_1, s_2, ..., s_n ($1 \le s_i \le 10^6$).

The third line contains n integer costs c_1, c_2, ..., c_n ($1 \le c_i \le 10^4$).

The fourth line contains one integer q: the number of knapsacks ($1 \le q \le 10^5$).

The fifth line contains q integer capacities x_1, x_2, ..., x_q of knapsacks ($1 \le x_i \le 10^6$).

Being nice to you, the jury has already sorted objects in non-increasing order by their sizes.

You should not change the given order.

Output

For each knapsack, print the total cost of objects inside it on a separate line.

You should solve the problem independently for each knapsack.

Example

greedy.in	greedy.out
5 5 4 4 2 1 10 1 100 20 40 3 4 8 100	1 70 171

Problem G. Matching

Input file:	`matching.in`
Output file:	`matching.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Consider an undirected graph with n vertices and m edges. Your task is to find a perfect matching in this graph and check if it is unique.

Input

The first line contains two integers n and m ($1 \le n \le 3000$, $0 \le m \le 10\,000$). Each of the next m lines contains a pair of integers each: ends of the corresponding edge. Vertices are labeled by integers from 1 to n.

It is guaranteed that no edge connects a vertex with itself and each pair of vertices is directly connected by at most one edge.

Output

Output "`NULLUS`" if there is no perfect matching or it is not unique. Otherwise output "`42`", and on the next $\frac{n}{2}$ lines, print pairs of integers: ends of edges from perfect matching. You may output edges and ends of each edge in any order.

Examples

matching.in	matching.out
4 3 1 2 2 3 3 4	42 4 3 1 2
4 4 1 2 2 3 3 4 4 1	NULLUS

Problem H. Matrix

Input file:	`matrix.in`
Output file:	`matrix.out`
Time limit:	1 second (*2 seconds for Java*)
Memory limit:	256 mebibytes

The White Rabbit woke up in a matrix with w columns and h rows ($1 \le w, h \le 2000$) at point (s_x, s_y). The red pill is located at point (e_x, e_y). The Rabbit wants to reach it.

The matrix consists of lowercase English letters. A *path* is a non-empty sequence of cells where each two neighboring cells share a side. In particular, this means that the neighboring cells of the path must be different. Let the *string of a path* be the sequence of letters on cells along that path (including start and finish cells).

Your goal is to find a path to the pill (it can pass through one cell multiple times) such that the string on that path is lexicographically the least possible. Obviously, all such paths start with a letter written in (s_x, s_y) and end with a letter written in (e_x, e_y).

Input

The first line contains two integers w and h: the dimensions of the matrix ($1 \le w, h \le 2000$). Each of the next h lines contains w lowercase English letters (without spaces). The last two lines contain coordinates s_x, s_y, e_x and e_y ($1 \le s_x, e_x \le w$, $1 \le s_y, e_y \le h$). Here, $(1, 1)$ is the leftmost cell of the top line and (w, h) is the rightmost cell of the bottom line.

Output

Output "`BLUEPILL`" if there is no lexicographically smallest path. Otherwise, output the sequence of letters `L`, `R`, `U`, `D` representing the path. Each letter corresponds to one move of the rabbit in the respective direction: `L` stands for left, `R` for right, `U` for up and `D` for down.

If there are several such paths, output any one of them.

Examples

matrix.in	matrix.out
4 3 aaaa bbba zaaa 1 3 1 1	RRRUULLL
3 3 aaa aaa aaz 1 1 3 3	BLUEPILL

Explanation

In the first example, the string on the path is "`zaaaaaaaa`". It is lexicographically the least possible string on a path from $(1, 3)$ to $(1, 1)$.

Problem I. Scheduling

Input file:	`scheduling.in`
Output file:	`scheduling.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Vasya is a film director. Right now, he needs to organize casting for his new film. He has n roles in the film. For i-th role, he has a_i candidates. No candidate claims two or more roles.

The selection process consists of several steps. On each step, Vasya chooses one candidate for each role and checks how they play together. Due to his aesthetic needs, Vasya wants to organize schedule in such a way that for each d candidates for different roles, there will be at least one step in which they all participate.

Vasya has little time, so the schedule must consist of not more than m steps. But this film is very important for him, so m is sufficiently large (the exact constraints are given in the input format description).

And, of course, you were asked to help him in the scheduling selection process. Fortunately, Vasya is very lucky, so it is guaranteed that at least one solution exists.

Input

The input consists of exactly two lines. The first line contains three integers n, m and d ($1 \le n \le 15$, $1 \le m \le 200\,000$, $1 \le d \le \min(4, n)$). The second line contains n integers a_1, a_2, ..., a_n ($15 \le a_i \le 200\,000$). It is guaranteed that $m \ge 2 \cdot \max(a_1, a_2, \ldots, a_n)^d$.

Output

On the first line, output the number of steps in your scheduling. After that, output the description of steps, one step per line. The i-th of these lines must contain exactly n integers $b_{i,1}$, $b_{i,2}$, ..., $b_{i,n}$: the numbers of candidates for roles 1, 2, ..., n which participate in i-th step. Each of these integers must satisfy $1 \le b_{i,j} \le a_j$.

Example

scheduling.in	scheduling.out
3 30 1 15 15 15	15 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6 7 7 7 8 8 8 9 9 9 10 10 10 11 11 11 12 12 12 13 13 13 14 14 14 15 15 15

Problem J. Pie Eating Tournament

Input file:	`tournament.in`
Output file:	`tournament.out`
Time limit:	4 seconds
Memory limit:	256 mebibytes

Petya is a famous competitive pie eater. He participates in lots of tournaments. His friend Vasya is a not-so-famous pie eater, and he is very upset by the fact that Petya is often ahead. So he asked his friend Oleg to calculate statistics in such a way that Vasya was ahead of Petya most times.

We assume that the pie eating tournaments are held in following way: each competitor plays one match against each other competitor, and in each match, one competitor wins, and the other one loses. We say that competitor A can be placed ahead of competitor B if there exists such a sequence of competitors P_1, P_2, ..., P_m that for each $i = 1 \ldots m - 1$, competitor P_i wins the match against P_{i+1}, and additionally, $P_1 = A$ and $P_m = B$.

Now Vasya and Oleg are interested in the number of different possible tournaments with N participants in which Vasya can be placed ahead of Petya. Let us assign numbers to competitors such that Vasya has number 1 and Petya has number 2. Vasya and Oleg consider two tournaments different if there are two numbers i and j such that in the first tournament, competitor i wins the match against competitor j, and in the second one, he loses that match.

Input

The first line of input contains a prime integer P ($10^8 \le P \le 10^9$). Next lines describe one or more test cases. Each of them contains one integer N ($2 \le N \le 4000$). There are no more then 3999 tests in one input file.

Output

For each test case, print the answer modulo P.

Example

tournament.in	tournament.out
948594971 2 3 4	1 5 48

Problem K. Tree

Input file:	`tree.in`
Output file:	`tree.out`
Time limit:	4 seconds (*6 seconds for Java*)
Memory limit:	256 mebibytes

A rooted tree with n vertices is given. Vertex number 1 is the root of the tree. Each vertex has a value val_i associated with it. Calculate the number of pairs of vertices (A, B) such that A is an ancestor of B and $val_A + val_B \le S$. A vertex is not considered an ancestor of itself.

Input

The first line of input contains two integers n and S ($2 \le n \le 10^6$, $1 \le S \le 10^6$). The second line contains $n-1$ integers: parents of vertices 2, 3, ..., n. The third line contains n integers val_1, val_2, ..., val_n ($1 \le val_i \le 10^6$).

Output

Print one integer: the required number of pairs.

Example

tree.in	tree.out
5 10 1 1 2 3 5 6 1 4 1	5

Explanation

In the example above, the pairs are $(1, 3)$, $(1, 4)$, $(1, 5)$, $(2, 4)$ and $(3, 5)$.