Problem A
AND Permutation
Time Limit: 2 second(s)
Memory Limit: 2G

You are given a sequence of n distinct nonnegative integers $a_1, a_2, \ldots, a_n$.

For the given sequence, it is guaranteed that for all nonnegative numbers x, if there is some i such that $a_i \ \& \ x = x$, then there is a j such that $a_j = x$. Here, $\&$ refers to the bitwise *AND* operator.

Find a permutation $b_1, b_2, \ldots, b_n$ of $a_1, a_2, \ldots, a_n$ such that $b_i \ \& \ a_i = 0$ for all i. If there are multiple solutions, find any such permutation. It is guaranteed that a solution always exists.

Input

The first line of input contains an integer n ($1 \leq n < 2^{18}$), which is the number of integers in the permutation.

Each of the next n lines contains an integer a_i ($0 \leq a_i < 2^{60}$), which is the input sequence, in order of i. All of the a_i's are guaranteed to be distinct. For all nonnegative numbers x, if there is some i such that $a_i \ \& \ x = x$, then there is a j such that $a_j = x$.

Output

Output n lines, each containing a single integer, which are the b_i's, in order of i.

Sample Input 1	Sample Output 1
6 0 1 4 5 2 6	4 6 0 2 5 1

This page is intentionally left blank.

Problem B
Apple Orchard
Time Limit: 15 second(s)
Memory Limit: 2G

Farmer John has many apple trees on his farm. Each apple tree has a circular area that provides shade during the hot summer. Farmer John is creating a pen for his cows and has several locations in mind. For each fenced off area he wants to know the percentage of that proposed area that is shaded.

Each proposed fenced off region is rectangular in shape, axis aligned, and specified by its lower left corner and the width and height of the region. Calculate the percentage of shaded area for each proposed fenced off rectangle.

Input

The first line of input contains two integers n ($1 \leq n \leq 3{,}000$) and q ($1 \leq q \leq 3{,}000$), where n is the number of apple trees in Farmer John's orchard, and q is the number of rectangular fenced regions he wishes to test.

Each of the next n lines contains three integers x, y ($-10^6 \leq x, y \leq 10^6$) and r ($1 \leq r \leq 10^6$). Each line describes the circular shaded area of a tree, where (x, y) is its center and r is its radius. Note that the trees can have very twisted trunks, so it is possible for two shaded areas to have the same center, or even be identical.

Each of the next q lines contains four integers x, y ($-10^6 \leq x, y \leq 10^6$), w and h ($1 \leq w, h \leq 10^6$). Each line describes a rectangular region Farmer John wishes to test. The rectangle has a diagonal from (x, y) to $(x + w, y + h)$.

Output

Output q lines, each containing a single real number, which is the percentage of that rectangle which is shaded, on a scale from 0 to 100. Output the percentages for the rectangles in the order that they appear in the input. Each value should lie within a relative or absolute error of 10^{-5} of the judges' answer.

Sample Input 1	Sample Output 1
2 2 0 0 3 2 1 4 0 0 3 3 -3 -3 6 6	100 89.53678472917

Sample Input 2	Sample Output 2
4 3 -1 -1 3 1 -1 3 -1 1 3 1 1 3 -4 -4 8 8 -1 -4 2 8 -3 -1 12 3	87.2221423775645 98.5869913729161 57.8623304576387

Problem C
Cleaning Robot
Time Limit: 8 second(s)
Memory Limit: 2G

A company wishes to purchase a square-shaped cleaning robot to clean a rectangularly shaped room. Some parts of the room are obstructed.

There are different robots of different sizes. Each robot can move horizontally and vertically in the room if no part of the robot intersects an obstruction. They are incapable of changing orientation, so movements are always axis-aligned. Larger robots will get the job done faster, but are more likely to be hindered by obstructions. The robot must always remain fully in the room with no portion extending past the edges of the rectangle.

What is the largest robot the company can buy that will be able to clean all the squares of the room not occupied by obstructions?

Input

The first line of input contains three integers n, m ($3 \leq n, m$ and $n \cdot m \leq 5 \cdot 10^6$) and k ($0 \leq k < n \cdot m$, $k < 10^6$), where n and m are the dimensions of the room in inches, and k is the number of obstructions.

Each of the next k lines contains two integers i and j ($1 \leq i \leq n, 1 \leq j \leq m$). This specifies that the one-inch square at (i, j) is obstructed. All obstructed squares are distinct.

Output

Output a single integer, which is the maximum length of one side of the largest square-shaped robot that could clean the entire room, or -1 if no such robot could clean the entire room.

Sample Input 1	Sample Output 1
10 7 1 8 3	2

This page is intentionally left blank.

Problem D
Contest Construction
Time Limit: 1 second(s)
Memory Limit: 2G

The ICPC NAC staff have written a number of problems and wish to construct a problem set out of them. Each problem has a positive difficulty rating.

A contest has a *Nice* difficulty distribution if, when the difficulties of the problems are sorted in ascending order, every problem after the second has a difficulty rating less than or equal to the sum of the difficulty ratings of the two problems immediately preceding that problem.

Given the difficulty ratings of a set of problems and the number of problems desired for the set, count the number of problem sets with *Nice* difficulty distributions. Two problem sets are distinct if and only if there is some problem in one problem set but not in the other. (Specifically, note that two problem sets are the same even if the problems are permuted.)

Input

The first line of input contains two integers n ($3 \leq n \leq 50$) and k ($3 \leq k \leq 18, k \leq n$), where n is the number of problems the judges have written, and k is the number of problems they wish to put in the problem set.

Each of the next n lines contains a single integer d ($1 \leq d \leq 10^9$). These are the problem difficulties.

Output

Output a single integer, which is the number of possible problem sets with *Nice* difficulty distributions.

Sample Input 1	Sample Output 1
5 4 2 1 4 3 5	4

Sample Input 2	Sample Output 2
8 5 1 2 3 5 8 13 21 34	4

Problem E
Ketek Counting
Time Limit: 4 second(s)
Memory Limit: 64M

Define a *Ketek* to be a sentence that reads the same forwards and backwards, by word. For example, 'fall leaves after leaves fall' is a *Ketek* since the words in reverse order are the same as the original order.

Given a string consisting of lower-case letters and the character '`?`', count the number of distinct *Keteks* you can make by replacing every '`?`' with lower-case letters (one letter per '`?`'), and optionally adding spaces between any letters. Note that a *Ketek* cannot contain any `?`'s; they all must be replaced exclusively by lower-case letters.

For example, if we start with the string '`ababa`', we can form 3 different *Keteks*: '`ababa`', '`a bab a`' and '`a b a b a`'.

If we start with the string '`?x?z`' instead, we can form 703 different *Keteks*:

- There are $26^2 = 676$ ways to replace the `?`'s and form a one-word *Ketek*.
- Add spaces to form '`? x? z`'. There are 26 ways to form a *Ketek* (the first '`?`' must be `z`; the other can be any lower-case letter).
- Add a space to form '`?x ?z`'. There is no way to form a *Ketek*.
- Add spaces to form '`? x ? z`'. There is one way to form a *Ketek* (the first '`?`' must be `z`; the second must be `x`).

The total is $676 + 26 + 0 + 1 = 703$.

Two *Keteks* are different if they have a different number of words, or there is some word index where the words are not the same.

Input

The single line of input contains a string s ($1 \leq |s| \leq 30{,}000$), which consists of lower-case letters (‘a’–‘z’) and the character ‘?’.

Output

Output the number of distinct *Keteks* that can be formed by replacing the ?’s with lower-case letters and adding spaces. Since this number may be large, output it modulo $998{,}244{,}353$.

Sample Input 1	Sample Output 1
ababa	3

Sample Input 2	Sample Output 2
?x?z	703

Problem F
Mountainous Palindromic Subarray
Time Limit: 2 second(s)
Memory Limit: 2G

An array is *Mountainous* if it is strictly increasing, then strictly decreasing. Note that *Mountainous* arrays must therefore be of length three or greater.

A *Subarray* is defined as an array that can be attained by deleting some prefix and suffix (possibly empty) from the original array.

An array or subarray is a *Palindrome* if it is the same sequence forwards and backwards.

Given an array of integers, compute the length of the longest *Subarray* that is both *Mountainous* and a *Palindrome*.

Input

The first line of input contains an integer n ($1 \leq n \leq 10^6$), which is the number of integers in the array.

Each of the next n lines contains a single integer x ($1 \leq x \leq 10^9$). These values form the array. They are given in order.

Output

Output a single integer, which is the length of the longest *Mountainous Palindromic Subarray*, or -1 of no such array exists.

Sample Input 1	Sample Output 1
8 2 1 2 3 2 1 7 8	5

Sample Input 2	Sample Output 2
5 2 5 8 7 2	-1

Problem G
Permutation CFG
Time Limit: 4 second(s)
Memory Limit: 2G

Consider a permutation of the integers 1 to n. Now, consider each number 1 through n to be a non-terminal in a *Context-Free Grammar* (CFG). Each number k expands a list of the integers from 1 to k in the order of the permutation. For example, if $n = 4$ and the permutation is 1 4 3 2:

$1 \implies 1$
$2 \implies 1\ 2$
$3 \implies 1\ 3\ 2$
$4 \implies 1\ 4\ 3\ 2$

Now, consider a process of starting with n, and at each step, applying these rules to create a new list of integers. In the above example, at the first step:

$$\overbrace{1\ 4\ 3\ 2}^{4}$$

At the second step:

$$\overbrace{1}^{1}\ \overbrace{1\ 4\ 3\ 2}^{4}\ \overbrace{1\ 3\ 2}^{3}\ \overbrace{1\ 2}^{2}$$

At the third step:

$$\overbrace{1}^{1}\ \overbrace{1}^{1}\ \overbrace{1\ 4\ 3\ 2}^{4}\ \overbrace{1\ 3\ 2}^{3}\ \overbrace{1\ 2}^{2}\ \overbrace{1}^{1}\ \overbrace{1\ 3\ 2}^{3}\ \overbrace{1\ 2}^{2}\ \overbrace{1}^{1}\ \overbrace{1\ 2}^{2}$$

Given a permutation, a number of steps, and a list of queries asking for the number of occurrences of a particular integer in a prefix of the list created by the process, answer all of the queries.

Input

The first line of input contains three integers, n ($2 \leq n \leq 10^5$), s ($1 \leq s \leq 5$) and q ($1 \leq q \leq 2 \cdot 10^5$), where n is the size of the permutation, s is the number of steps to apply the process, and q is the number of queries.

Each of the next n lines contains a single integer p ($1 \leq p \leq n$). This is the permutation, in order. All of the values of p will be distinct.

Each of the next q lines contains two integers k ($1 \leq k \leq n$) and a ($1 \leq a \leq 10^9$, a will not exceed the length of the final list). This is a query for the number of occurrences of the integer k in the first a elements of the list created by the process.

Output

Output q lines, each with a single integer, which are the answers to the queries in the order that they appear in the input.

Sample Input 1	Sample Output 1
4 3 6 1 4 3 2 1 6 2 20 4 1 3 5 2 9 1 16	3 6 0 1 2 8

Problem H
Special Cycle
Time Limit: 7 second(s)
Memory Limit: 2G

You are given a simple undirected graph with no self-loops or multiple edges. Some of the edges are marked as *Special*.

Your task is to find a simple cycle where, for each *Special* edge, that edge either belongs to the cycle or neither of its endpoints touch the cycle. The cycle is not allowed to repeat vertices. Output any solution, or report that none exist.

Input

The first line of input contains three integers n ($2 \leq n \leq 150$), m ($1 \leq m \leq \frac{n \cdot (n-1)}{2}$) and k ($1 \leq k \leq m$), where n is the number of nodes in the graph, m is the number of edges, and k is the number of edges that are *Special*. The nodes are numbered 1 through n.

Each of the next m lines contains two integers a and b ($1 \leq a < b \leq n$), denoting an undirected edge between nodes a and b. All edges are distinct. The first k edges are the *Special* edges.

Output

Output an integer denoting the length of the found cycle on one line. On subsequent lines, output the vertices of the cycle in order around the cycle, one per line. If no such cycle exists, simply output -1.

Sample Input 1

```
8 10 3
1 2
4 5
7 8
2 3
3 4
1 4
5 6
6 7
5 8
3 8
```

Sample Output 1

```
8
1
4
5
6
7
8
3
2
```

Sample Input 2

```
4 6 3
1 2
1 3
1 4
2 3
3 4
2 4
```

Sample Output 2

```
-1
```

Problem I
The King's Guards
Time Limit: 1 second(s)
Memory Limit: 2G

In a certain kingdom, the king wants to protect his citizens by deploying guards. He has recruited a number of guards, and has outfitted them with heavy armor for protection from bandits, foreign knights, and other ne'er-do-wells. His guards are tough, but unfortunately they aren't very bright and will attack anyone wearing armor, even each other!

The kingdom consists of a number of villages, connected by roads. All of the roads are of poor quality. Some are muddy, some have rickety bridges. None of them can support a guard in full armor. So, the king must decide which roads to improve so that his guards can reach the entire kingdom. The roads are bidirectional. Each guard can only be deployed to a single village in a certain subset of the kingdom's villages.

The king needs to minimize the cost of improving roads, while satisfying several other constraints:

- Every guard must be deployed; none must be left out.
- Every guard must be deployed in their subset of villages.
- Every village must be reachable by exactly one guard. If two guards can reach each other, they'll fight.

Help the king determine the minimum cost of improving the roads of his kingdom while satisfying the above constraints.

Input

The first line of input contains three integers n ($1 \leq n \leq 300$), r ($0 \leq r \leq \frac{n \cdot (n-1)}{2}$) and g ($1 \leq g \leq n$), where n is the number of villages, r is the number of roads, and g is the number of guards. The villages are numbered 1 through n.

Each of the next r lines contains three integers a, b ($1 \leq a < b \leq n$) and c ($1 \leq c \leq 1{,}000$). Each line describes a road between village a and village b, costing c to improve. The roads are bidirectional; a guard can go from a to b or from b to a. Every pair of villages has at most one road between them.

Each of the next g lines starts with an integer k ($1 \leq k \leq n$), and then contains k integers v ($1 \leq v \leq n$). Each line describes the villages comprising the subset where one particular guard may be placed. The subsets may overlap.

Output

Output a single integer, which is the minimum cost the king must pay to improve enough roads so that every village is reachable by exactly one guard, and every guard is deployed. Output -1 if it isn't possible to deploy the guards in a way that satisfies all of the constraints.

Sample Input 1	Sample Output 1
5 6 2 1 2 1 1 3 4 2 4 2 2 5 5 3 4 7 4 5 3 2 1 2 2 2 4	8

Problem J
The Paladin
Time Limit: 1 second(s)
Memory Limit: 2G

A Paladin, a warrior adept at holy magic, needs your help forming spells. Being a Paladin, every spell must also be a *Palindrome*, meaning that it is the same string when read forwards and backwards. The cost of constructing a new spell is based on rune pair costs (costs of adjacent letters) that occur in the final spell. Rune pairs not presented in the input are not allowed in the final spell.

The total cost of a spell is the sum of all rune pair costs in the spell for each time the pair occurs in the spell. For example, if the spell is **`abacaba`**, then the cost is the sum of the costs of **`ab`** + **`ba`** + **`ac`** + **`ca`** + **`ab`** + **`ba`**.

Determine the smallest cost to make a new Paladin spell of exactly a given length.

Input

The first line of input contains two integers n ($1 \leq n \leq 676$) and k ($2 \leq k \leq 100$), where n is the number of rune pairs and k is the desired spell length in runes (i.e., letters).

Each of the next n lines contains a string s (s consists of exactly two lower-case letters, which may be the same or different) and an integer c ($1 \leq c \leq 100$), where the string s is a rune pair, and c is the cost of that rune pair. All rune pairs are distinct.

Output

Output a single integer, which is the smallest possible cost for constructing a spell of length k from the given runes, or -1 if it isn't possible.

Sample Input 1	Sample Output 1
5 9 ab 4 ba 1 bd 3 db 100 bc 4	20

This page is intentionally left blank.

Problem K
Token Game
Time Limit: 3 second(s)
Memory Limit: 2G

Alice and Bob are playing a game on board which is a 2-dimensional 300×300 grid. The board is subdivided into cells. Each cell can be uniquely identified by two integers representing (x, y) coordinates, each in the range from 1 to 300.

Two tokens are on the board on distinct cells. Alice starts the game. On each player's turn, that player chooses one of the tokens, chooses one of the coordinates of the cell it's on, and reduces that coordinate by some positive amount. The moved token cannot jump over or occupy the same space as the other token. The token must also remain on the board (so both of its coordinates need to stay positive). The first player unable to make a move loses. Note that both players can move either token.

You are given the starting configuration of a number of games. For each of the games, compute the number of initial winning moves available to Alice.

Input

The first line of input contains a single integer n ($1 \leq n \leq 10^5$), which is the number of games to analyze.

Each of the next n lines contains four integers x_1, y_1, x_2 and y_2 ($1 \leq x_1, x_2, y_1, y_2 \leq 300$, and either $x_1 \neq x_2$ or $y_1 \neq y_2$ holds). This represents the starting configuration of one game, with the tokens at cells (x_1, y_1) and (x_2, y_2).

Output

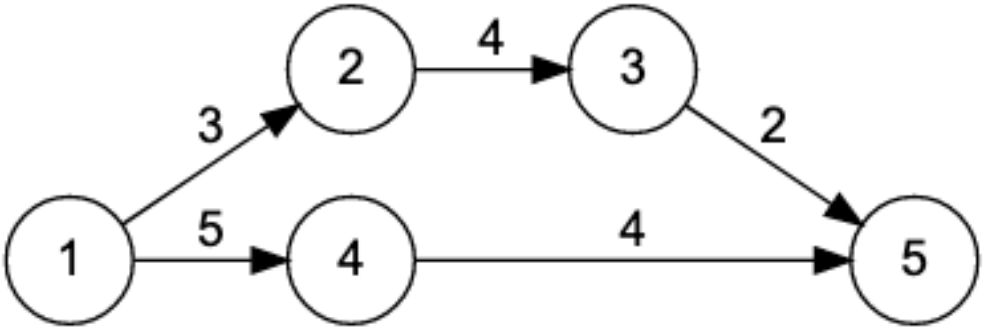
Output n lines. On each line, output a single integer, which is the number of initial winning moves available to Alice for one of the input games. Output them in the order of the input.

Sample Input 1	Sample Output 1
5 6 6 6 3 6 6 2 2 1 6 3 1 3 6 1 3 6 3 1 5	3 0 1 1 0

Problem L
TraveLog
Time Limit: 4 second(s)
Memory Limit: 2G

After a long time apart, Alice and Bob have reunited. They live in a country with n cities, creatively named city 1 to city n. Bob drove from his home in city 1 to Alice's home in city n.

When Alice asked Bob which path he took, he was stunned to find he didn't remember. Bob is efficient, and drove without stopping, and knows there is no path faster than the one he took. He also has a brand new National Adventuring Company (NAC) TraveLog! Every time Bob drives through a city, the TraveLog records the time between when he left city 1 and when he arrived in the current city.



In the above layout, there are two possible fastest paths Bob can take from city 1 to city n: $1 \to 2 \to 3 \to 5$ or $1 \to 4 \to 5$. Both paths take a total of 9 units of time. The first path would have a TraveLog of $0, 3, 7, 9$, and the second would have a TraveLog of $0, 5, 9$.

Unfortunately, the memory in Bob's TraveLog is corrupted. Bob thinks that some of the times are gone, and the remaining times are shuffled arbitrarily. Given what remains of the TraveLog, can you reconstruct Bob's path?

Input

The first line of input contains three integers n ($2 \le n \le 2 \cdot 10^5$), m ($1 \le m \le 3 \cdot 10^5$) and d ($1 \le d \le n$), where n is the number of cities in the country, m is the number of one-way roads between those cities, and d is the number of times remaining in Bob's corrupted TraveLog. The cities are identified by number, 1 through n. Bob lives in city 1, and Alice lives in city n.

Each of the next m lines contains three integers u, v ($1 \le u, v \le n, u \ne v$) and h ($1 \le h \le 10^6$). Each line describes a one-way road from city u to city v that takes h units of time to traverse. There

is guaranteed to be at least one path from city 1 to city n. There may be several roads between any given pair of cities.

Each of the next d lines contains a single integer t ($0 \leq t \leq 10^{18}$). This is what remains of Bob's TraveLog. Each line is a record of the amount of time Bob took driving from city 1 to another city on his path. These values are guaranteed to be distinct.

Output

The output format depends on the number of paths consistent with Bob's TraveLog.

- If there is no path consistent with Bob's TraveLog, output 0.
- If multiple paths are consistent with Bob's TraveLog, output 1.
- Otherwise, on the first line, output the number of cities on Bob's path. On subsequent lines, output the cities Bob visited, one per line, in the order he visited them.

Sample Input 1	Sample Output 1
5 5 2 1 2 3 2 3 4 3 5 2 1 4 5 4 5 4 5 9	3 1 4 5

Sample Input 2	Sample Output 2
6 8 2 1 2 1 2 3 2 3 6 8 1 4 3 4 5 4 5 6 4 5 2 7 1 6 13 0 3	1

Sample Input 3	Sample Output 3
2 1 1 1 2 10 5	0

This page is intentionally left blank.

Problem M
You be The Judge, Again
Time Limit: 2 second(s)
Memory Limit: 2G

You are a judge, again! The contest you're judging includes the following problem:

> "You have one L-shaped triomino of each of $\frac{4^n-1}{3}$ different colors. Tile a 2^n by 2^n grid using each of these triominos such that there is exactly one blank square and all other squares are covered by exactly one square of such a triomino. All triominos must be used."

Your team is to write a checker for this problem. Validation of the input values and format has already taken place. You will be given a purported tiling of a 2^n by 2^n grid, where each square in the grid is either 0 or a positive integer from 1 to $\frac{4^n-1}{3}$ representing one of the colors. Determine if it is, indeed, a covering of the grid with $\frac{4^n-1}{3}$ unique triominos and a single empty space.

L-shaped triominos look like this:

Input

The first line of input contains a single integer n ($1 \leq n \leq 10$), which is the n of the description.

Each of the next 2^n lines contains 2^n integers x ($0 \leq x \leq \frac{4^n-1}{3}$), where 0 represents an empty space, and any positive number is a unique identifier of a triomino.

Output

Output a single integer, which is 1 if the given grid is covered with $\frac{4^n-1}{3}$ unique triominos and a single empty space. Otherwise, output 0.

Sample Input 1	Sample Output 1
2 1 1 2 2 1 3 3 2 4 4 3 5 4 0 5 5	1

Sample Input 2	Sample Output 2
1 1 1 1 1	0