Problem A
Contact Tracing
Time Limit: 1 Second(s)

A novel infectious disease has started spreading through the population. You are tasked with figuring out who might be infected in order to get them to quarantine. The behavior of the disease among members of the population is as follows:

- When a person comes into contact with someone who is infectious, they may (but do not always) catch the disease. If they catch the disease, they are not infectious for the rest of that day, so they will not spread it to other people they come into contact with that day.
- When a person catches the disease, they are infectious starting the day after they caught the disease.
- Starting two days after they caught the disease, they are symptomatic; as a result, they will quarantine themselves and not come into contact with any other people from then on.

You know that on day 0, a single Patient Zero (of unknown identity) caught the disease. On day 1, they were infectious and potentially came into contact with people, potentially infecting those people. On day 2, Patient Zero became symptomatic and therefore stopped coming into contact with other people, but any people who were infected on day 1 could potentially spread the disease to the people they come into contact with.

Today is day k, and it is the end of the day. You have access to a list of all pairs of people who came into contact with each other on each day, including today. If you can identify all people who could be infectious but not symptomatic tomorrow (because they caught the disease today) and tell them to quarantine, you can halt the outbreak, because all people who are symptomatic tomorrow will quarantine themselves anyway. What is the minimum number of people you need to tell to quarantine to be sure that you halt the outbreak?

Input

The first line of input contains three integers n ($1 \leq n \leq 1{,}000$), k ($1 \leq k \leq 10$), and c ($0 \leq c \leq 1{,}000$), where n is the number of people, k is the number of days since Patient Zero became infected, and c is the number of contacts that occurred between people.

Each of the next c lines contains three space-separated integers a, b ($1 \leq a < b \leq n$), and d ($1 \leq d \leq k$), denoting that people a and b came into contact with each other on day d. All of these c lines will be distinct. There will be at least one person with no contacts after day 1.

Output

Output a line with a single integer x denoting the minimum number of people you must tell to quarantine beginning on day $k + 1$. Then, output x lines listing the people who must quarantine, one per line, in ascending order.

Explanation of Sample Input 1

In Sample Input 1, Patient Zero could be person 1 or person 4. Persons 2 and 3 can be eliminated, because they had contact on day 2, when Patient Zero would be symptomatic and quarantined.

Suppose person 1 is Patient Zero. Person 1 may have infected person 4 on day 1. Person 1 is symptomatic on day 2, so they begin quarantining. Therefore, there is no chain of contact from person 1 (Patient Zero) to persons 2 or 3, so person 2 and person 3 are safe and do not need to be quarantined. If person 1 infected person 4 on day 1, then person 4 will be symptomatic tomorrow (day 3), so there is no need to contact them because they will isolate on their own. Therefore if Patient Zero is person 1, no one needs to be notified.

Now suppose person 4 is Patient Zero. Person 4 may have infected person 1 on day 1; using the same logic as above, there is no need to notify person 1 or person 4 to quarantine on day 3. This leaves persons 2 and 3, which requires us to consider multiple infection patterns.

If person 4 infected both persons 2 and 3 on day 1, then both of them will be symptomatic and self-quarantine starting on day 3, so we do not need to notify them.

Suppose instead person 4 infected person 2 but NOT person 3 on day 1. (Remember that transmission is not guaranteed.) In this case, person 2 could go on to infect person 3 on day 2, and person 3 would not yet be symptomatic on day 3, so we would need to notify person 3 to quarantine.

Similarly, if person 4 infected person 3 but NOT person 2 on day 1, we would need to notify person 2 to quarantine.

Therefore to be sure we can halt the outbreak, we need to notify both person 2 and person 3 to quarantine. This is guaranteed to stop the outbreak, whether person 1 or person 4 is Patient Zero.

Sample Input 1

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

Sample Output 1

```
2
2
3
```

Problem B
Cookie Cutter
Time Limit: 8 Second(s)

Everyone loves chocolate chip cookies! Which unfortunately means sharing sometimes. In this case, you have generously agreed to split a square chocolate chip cookie with your friend.

Because it is your cookie, you get to choose how to cut the cookie, and which piece to give to your friend. You can split the cookie along any straight line that passes through the cookie; the line need not be axis-aligned.

You know the location of all chocolate chips in the cookie. Because you prefer a cookie that is dense with chocolate chips, you want to optimize your cut to produce the best possible split. You accomplish this by maximizing the difference between the fraction of chocolate chips in your piece and the fraction of cookie area in your piece.

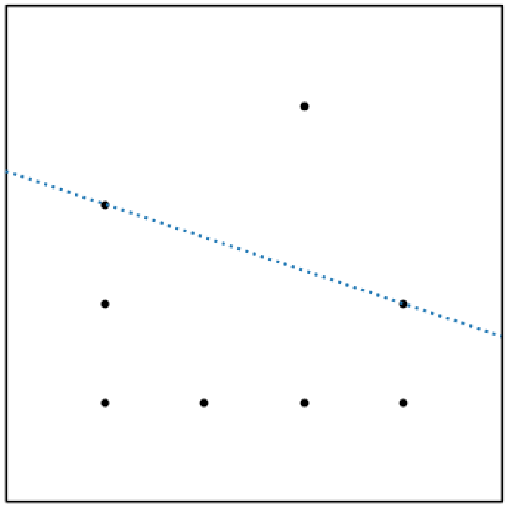


Illustration of Sample Case 1.

Input

The first line of input contains two space-separated integers n ($2 \le n \le 10{,}000$) and m ($1 \le m \le 3{,}000$), where n is the side length of the square cookie, and m is the number of chocolate chips in the cookie.

The next m lines each contain two space-separated integers x and y ($0 < x, y < n$) defining the location of a chocolate chip in the cookie. All chocolate chip locations are distinct. If a chocolate chip lies exactly on the cut, you can decide which piece of the cookie it goes to.

Output

Output a real number, which is the maximum possible value for $\frac{b}{m} - \frac{a}{n^2}$, where a is the area of cookie that you get, and b is the number of chocolate chips in your piece of the cookie. The answer is accepted with absolute or relative error at most 10^{-6}.

Sample Input 1

```
5 8
1 1
1 2
1 3
2 1
3 1
3 4
4 1
4 2
```

Sample Output 1

```
0.375
```

Problem C
Cram
Time Limit: 1 Second(s)

You want to compress a given text passage using backreferences. A *backreference* is a pair of numbers $[a, b]$ indicating that the next b characters of the string are the same as the b characters starting a characters back from the current position. The two strings may overlap, *i.e.*, a may be smaller than b.

Each backreference costs three bytes to encode, regardless of the number of characters represented by the backreference. String characters cost one byte each to encode.

For instance, the string

```
abcabcabcabc
```

has 12 characters. But the last nine can be represented as a backreference to the first nine, as follows:

```
abc[3,9]
```

The total cost of this encoded string is 6: 3 bytes for the string `abc`, and 3 bytes for the backreference.

Output the minimum cost to encode the text passage.

Input

The single line of input contains a string s, with $1 \le |s| \le 10^5$. This line of text consists of upper-case letters (‘`A`’–‘`Z`’), lower-case letters (‘`a`’–‘`z`’), and spaces. There will not be any spaces at the beginning or end of the line, and no space character will be adjacent to another space character.

Output

Output a single integer, which is the minimum cost to represent the input string using backreferences.

Sample Input 1	Sample Output 1
`abcabcabcabc`	`6`

Sample Input 2	Sample Output 2
`aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa`	`4`

Sample Input 3

```
A man a plan a canal Panama
```

Sample Output 3

```
25
```

Problem D
Double Sort
Time Limit: 1 Second(s)

Given two integers n and m ($n \leq m$), you generate a sequence of n integers as follows:

1. First, choose n distinct integers between 1 and m, inclusive.
2. Sort these numbers in non-decreasing order.
3. Take the difference sequence, which transforms a sequence $a_1, a_2, a_3, \ldots$ into $a_1, a_2 - a_1, a_3 - a_2, \ldots$
4. Sort the difference sequence in non-decreasing order.
5. Take the prefix sums of the sorted difference sequence to get the final sequence. This transforms a sequence $b_1, b_2, b_3, \ldots$ into $b_1, b_2 + b_1, b_3 + b_2 + b_1, \ldots$

For example, with $n = 3$ and $m = 10$:

1. Suppose we initially chose $6, 2, 9$.
2. The sequence in order is $2, 6, 9$.
3. The difference sequence is $2, 4, 3$.
4. The sorted difference sequence is $2, 3, 4$.
5. The prefix sums of the sorted difference sequence are $2, 5, 9$.

Suppose you chose a uniformly random set of distinct integers for step 1. Compute the expected value for each index in the final sequence.

Input

The single line of input contains two integers n ($1 \leq n \leq 50$) and m ($n \leq m \leq 10{,}000$), where n is the size of the sequence, and all of the initial integers chosen are in the range from 1 to m.

Output

Output n lines. Each line contains a single real number, which is the expected value at that index of the final sequence. Each answer is accepted with absolute or relative error at most 10^{-6}.

Sample Input 1	Sample Output 1

`3 5`	`1`
	`2.3`
	`4.5`

Problem E
GCD Harmony
Time Limit: 4 Second(s)

Consider a tree with undirected edges, where each node has an integer value. Adjacent nodes are said to be *GCD-harmonic* if the greatest common divisor (GCD) of their values is strictly greater than 1.

You can modify the value of any tree node to any positive integer. The cost of this operation is equal to the new node value, regardless of the node's original value. You can change as many node values as needed, and node values do not need to be unique.

What is the minimum total cost to make every pair of adjacent nodes in the tree GCD-harmonic?

Input

The first line of input contains a single integer n ($2 \leq n \leq 5{,}000$), which is the number of nodes in the tree. Tree nodes are numbered from 1 to n.

Each of the next n lines contains an integer v ($1 \leq v \leq 100$). These are the initial values of the nodes (which are not guaranteed to be unique), in node number order.

Each of the next $n-1$ lines contains two integers a and b ($1 \leq a, b \leq n, a \neq b$), indicating a tree edge between nodes a and b. It is guaranteed that these edges form a tree.

Output

Output a single integer, which is the minimum total cost to make every pair of adjacent nodes in the tree GCD-harmonic.

Sample Input 1

```
6
5
6
3
4
9
12
1 2
1 3
1 4
1 6
3 5
```

Sample Output 1

```
6
```

Sample Input 2

```
3
1
2
3
3 1
2 3
```

Sample Output 2

```
4
```

Problem F
Leaderboard Effect
Time Limit: 3 Second(s)

You are in charge of a programming contest with a number of distinct problems and an overall time limit. The judges provide for each problem a reading time, a coding time, and a solution probability.

During the contest, teams can view a leaderboard that shows the accepted submissions for each team. Seeing the leaderboard can have an effect on teams' strategies! All teams will follow the following strategy when solving problems in the contest:

1. Choose to read a problem. To choose which problem to read, they first gather all the problems they haven't yet read (if they have read all problems, they will spend the rest of the contest doing nothing). If none of these problems have any solutions, they will choose to read one uniformly at random. Otherwise, they will choose a problem to read, randomly weighted by the number of solutions that it has. For example, if there are three problems **A**, **B**, **C** with 3, 1, and 0 solutions respectively, they will choose to read problem **A** with probability $\frac{3}{4}$, problem **B** with probability $\frac{1}{4}$, and problem **C** with probability $\frac{0}{4}$.
2. After choosing any given problem, they first read it. This always takes the given reading time for that problem. After they finish reading the problem, they will know if they can solve it or not. The team will be able to solve the problem with the given probability (and this takes no time to decide). If they can't solve the problem, the team goes back to step 1 and chooses another problem to read. Otherwise, they spend the given time coding the problem (even if there is not enough time left in the contest to finish) before going back to step 1 to choose another problem to read. Once a team finishes coding the chosen problem, the problem is solved (receiving judgement takes no time); each problem solution affects the leaderboard, and therefore the problems other teams will attempt.

At all moments of time, the leaderboard will first update with all the solutions that happened at that time, then teams will see the updated leaderboard.

Let $f(m, i)$ be the expected number of teams that will solve problem i if there are m teams participating in the contest. Let $g(i) = lim_{m\to\infty}[\frac{f(m,i)}{m}]$, which is the expected proportion of teams that will solve problem i for a large enough number of teams. Output $g(i)$ for all problems i in the set.

Input

The first line of input contains two integers n ($1 \leq n \leq 17$) and t ($1 \leq t \leq 100$), where n is the number of problems in the contest, and t is the time limit in minutes.

Each of the next n lines contains two integers r and c ($1 \leq r, c \leq t$) and one real number p ($0.0 \leq p \leq 1.0$). Each line describes a problem, where r is reading time in minutes, c is coding time in minutes, and p is the probability of solving the problem.

Output

Output n lines. Each line contains a single real number, which is the expected proportion of teams that will solve that particular problem, given a large number of teams. Output these proportions for the problems in the same order as the input. The answers are accepted with absolute or relative error at most 10^{-6}.

Sample Input 1

```
4 42
10 10 0.75
10 10 0.75
10 12 1
10 12 1
```

Sample Output 1

```
0.45625
0.45625
0.296875
0.296875
```

Sample Input 2

```
4 42
10 12 0.75
10 12 0.75
10 10 1
10 10 1
```

Sample Output 2

```
0.203125
0.203125
0.683238636363636
0.683238636363636
```

Sample Input 3

```
5 100
40 60 0.6
40 61 1
10 40 0.3
10 40 0.4
10 40 0.5
```

Sample Output 3

```
0.12
0
0.112628571428571
0.159739682539683
0.206444444444444
```

Problem G
Rafting Trip
Time Limit: 1 Second(s)

You are planning a rafting trip. The terrain can be viewed as a grid. Each cell is either land, or has part of a river flowing through it in one of the four directions: north, south, east, or west. Some land cells contain a sightseeing spot.

You can choose any river cell as the starting point of your rafting trip. Once your raft reaches a river cell (including the starting cell), it will follow the water direction of that cell and move to an adjacent cell or exit the grid.

You can visit a nearby sightseeing spot if your raft reaches a river cell that is adjacent to it, including the starting cell. (Cell adjacency includes horizontal and vertical neighbors, but not diagonal neighbors.) Each sightseeing spot can be visited at most once.

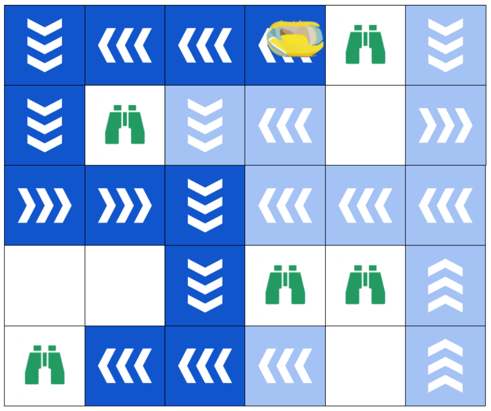


Illustration of the first sample case. The optimal rafting trip starts at the cell with the raft and visits 4 sightseeing spots (marked by binoculars). The river cells reached along the trip are highlighted in dark blue.

Your rafting trip stops when your raft moves onto a land cell, exits the grid, or enters a river cell that it has reached before. Note that if the raft ends at a land cell, you cannot visit the sightseeing spots adjacent to that land cell.

What is the maximum number of sightseeing spots you can visit in a single rafting trip if you choose your starting cell optimally?

Input

The first line of input contains two integers r and c ($2 \leq r, c \leq 500$); the terrain grid has r rows and c columns.

Each of the next r lines contains c characters describing one row of the terrain grid. A dot '.' denotes a land cell without a sightseeing spot. A hash '#' denotes a land cell that contains a sightseeing spot. River cells are denoted by '^' (north), 'v' (south), '>' (east), or '<' (west). There is at least one river cell in the grid.

Output

Output a single line with a single integer, which is the maximum number of sightseeing spots you can visit in a single rafting trip.

Sample Input 1

```
5 6
v<<<#v
v#v<.>
>>v<<<
..v##^
#<<<.^
```

Sample Output 1

```
4
```

Sample Input 2

```
4 5
>v<<.
^<..#
#...#
.#>^#
```

Sample Output 2

```
2
```

Problem H
Ranked Choice Spoiling
Time Limit: 1 Second(s)

Ranked choice voting, a way of determining the winner of an election, proceeds as follows:

1. Each candidate (identified in this problem as **`A`**, **`B`**, **`C`**, etc.) is placed on the ballot.
2. Each voter ranks the candidates from most to least favorite (e.g., **`B`** first, then **`A`**, then **`C`** last) and submits that ranking as their vote.
3. Once all votes have been received, the first place votes for each candidate remaining on the ballot are tallied. If there is a candidate with more than half of the first place votes, that candidate is declared the winner.
4. If no candidate has more than half of the first place votes, the candidate with the fewest first place votes is eliminated from the ballot, and the process returns to step 3 with the remaining candidates. (If there are multiple such candidates, the loser is the lexicographically last candidate, e.g., **`C`** is eliminated before **`B`**.)

Spoiling refers to an additional candidate **`Z`** entering a race, thereby causing the winner to switch from one existing candidate to another candidate (who is not **`Z`**). Proponents of ranked choice voting claim that this type of election is less vulnerable to spoiling than the more common plurality voting (the candidate with the most votes wins).

You are candidate **`A`** in an election against one or two other candidates, and you wish to prove them wrong by engineering a candidate **`Z`** to enter the election and spoil it in your favor. Suppose you knew every voter's rankings for the existing candidates, and you were capable of engineering a candidate **`Z`** such that you had full control over where each voter inserts **`Z`** into their rankings. For a given set of such rankings, is it possible to engineer a candidate **`Z`** to spoil the election in your favor?

Input

The first line contains two integers n ($1 \leq n \leq 1{,}000$) and k ($2 \leq k \leq 3$), where n is the number of voters and k is the number of existing candidates.

Each of the next n lines contains k space-separated capital letters (**`A`**, **`B`**, or **`C`**) indicating the rankings of each voter from first to last place. Each line will list each candidate exactly once (when $k = 2$, **`C`** is not present).

Output

Output a single integer, 1 if it is possible to create a candidate **Z** who spoils the election in favor of candidate **A** (or if **A** would already win the election), 0 otherwise.

Sample Input 1	Sample Output 1
5 2 B A A B A B B A B A	1

Sample Input 2	Sample Output 2
1 3 A B C	1

Sample Input 3	Sample Output 3
8 3 A B C B C A B C A B C A C B A C B A C B A C B A	0

Problem I
Reset
Time Limit: 7 Second(s)

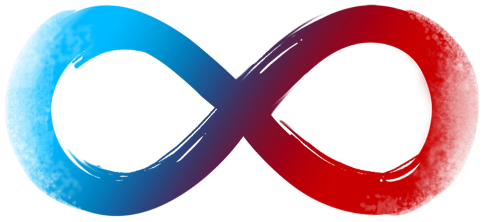


Image from pngimg.com

Li and Xiao are involved in a supernatural event in which a crisis will happen at a fixed time in the future. They must complete a number of tasks to prevent the crisis from happening. Once they start a task, they must finish it before switching to another task. Tasks can be completed in any order.

On their first attempt, there may not be enough time for Li and Xiao to complete all the tasks before the crisis hits. The crisis will happen if any task is not completed in time, and Li and Xiao will die. Fortunately, at the moment of their death, the world will be *reset*: time is rewound to zero and Li and Xiao can have another attempt. All task progress will be lost upon a reset.

Li and Xiao can choose to research a task and gain experience in the task. For each full second Li and Xiao spend researching a task (they cannot spend a partial second on research), their completion time for that task will be reduced by some number of seconds. When the time is reduced to zero, Li and Xiao can complete the task instantly. Upon a reset, their experience from task research remains intact, and in their next attempt they will be able to complete the tasks more quickly. However, there is a constraint that each task can be researched at most once in each attempt.

Witnessing the crisis repeatedly (and therefore dying repeatedly) is not fun. Li and Xiao therefore want to minimize the number of resets they go through before they eventually prevent the crisis.

Input

The first line of input contains two integers n ($1 \leq n \leq 2 \cdot 10^5$) and c ($1 \leq c \leq 10^9$), where n is the number of tasks that Li and Xiao must complete, and c is the number of seconds until the crisis occurs.

Each of the next n lines contains two integers t and d ($1 \leq d \leq t \leq 10^9$) which describe a task, where t is the initial time required to complete the given task. If Li and Xiao research the task for one second, the amount of time to complete it is reduced by d; if this would cause the task completion time to become negative, it is reduced to 0.

Output

Output a single integer, the minimum number of resets required to complete all tasks and prevent the crisis.

Sample Input 1	Sample Output 1

3 5
17 5
5 2
15 4
	3

Sample Input 2	Sample Output 2

2 1345
1344 1
10 10
	0

Problem J
Tic Tac Toe Counting
Time Limit: 3 Second(s)

Tic Tac Toe is a simple children's game. It is played on a 3×3 grid. The first player places an `X` in any of the 9 cells. The next player places an `O` in any of the remaining 8 cells. The players continue to alternate placing `X`s and `O`s in unoccupied cells until either a player gets three of their symbols in a row, or the grid is full. If either player gets three in a row, in any of the three rows, three columns or two diagonals, that player wins and the game ends.

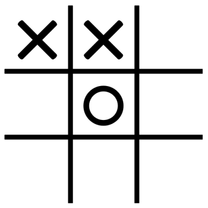
Fred and Sam are playing a game of Tic Tac Toe. In the middle of the game, Fred wonders: "How many games from this point in the game onward are winners for `X`? How many for `O`?" Two games are different if they have different sequences of moves, even if they result in the grid looking the same at any point.

Given a list of grids, determine first if they can be reached in a game of Tic Tac Toe, and if they can, how many games from that point on result in wins for `X`, and how many for `O`.

Input

The first line of input contains a single integer t ($1 \le t \le 10^5$), which is the number of test case grids.

Each of the next t lines contains a single string s ($|s| = 9$) which consists of only the characters '`.`', '`X`' and/or '`O`'. These are the test case grids. The first three characters represent the first row, the second three are the second row, and the last three are the last row, with '`.`' representing an empty cell. For example, the string `XX..O....` represents this grid:



Output

For each test case, output two space-separated integers. If the grid is not reachable in a game of Tic Tac Toe, output `-1 -1`. If the grid is reachable, output the number of games from that point on (including that grid) that are wins for `X`, followed by wins for `O`.

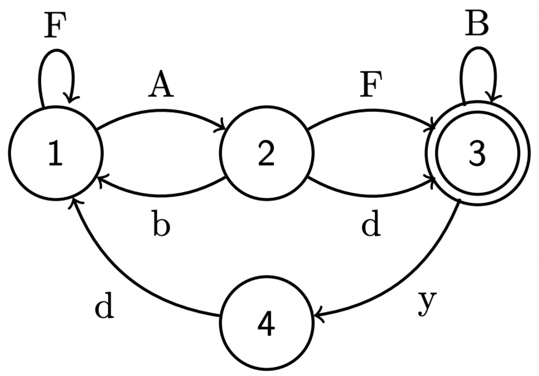
Sample Input 1

```
4
XX..O....
X...OX...
OOOX.X.X.
OOOXXX...
```

Sample Output 1

```
191 194
232 200
0 1
-1 -1
```

Problem K
Transparency
Time Limit: 6 Second(s)



The factory design from the first sample input. The packaging station is marked by a double circle.

You must audit a factory that produces a number of products. Each product is subject to its own taxation rules and regulations. Unfortunately, deciding which rules apply to a product is not obvious since it can be difficult to distinguish the products from one another. It is your job to investigate if the factory is being completely *transparent* in its production process by determining if the factory is capable of producing two different products that are indistinguishable from one another.

The final products are represented by sequences of uppercase letters (‘`A`’–‘`Z`’) and lowercase letters (‘`a`’–‘`z`’). Two final products are indistinguishable from one another if the products are equal after removing all lowercase letters. For example, `WabXYcz` is indistinguishable from `gWXfbY`, since after removing all lowercase letters we are left with `WXY` in both instances.

A factory is modelled as a set of stations S, a set of directed and labelled transition edges T, an initial station s_0, and a set of packaging stations P. The initial station is in the set of stations, i.e., $s_0 \in S$, and the set of packaging stations is a subset of S, i.e., $P \subseteq S$. A transition edge is defined by a 3-tuple (s_1, σ, s_2), where $s_1, s_2 \in S$ are stations, and σ is an uppercase or lowercase letter. The existence of the transition (s_1, σ, s_2) in T means that if in producing some product we are at station s_1, then appending σ to the product will leave us at station s_2. That is, there is a directed edge from station s_1 to station s_2, labelled σ.

A product can be produced by the factory if by starting at station s_0 there is a sequence of edges that can be followed, ending at a station in P, whose edge labels can be concatenated, in order, to create the product. The sequence of edges can be empty if $s_0 \in P$. For example, the following strings can all be produced by the factory described in the first sample input and illustrated in the figure: `AF`, `FAFB`, `AbFFAd`, `AdydAd`. Note that this is not an exhaustive list.

Each production transition can be used an unlimited number of times to make a product. It is guaranteed that for each station, s, and letter, σ, there is at most one directed edge from station s labelled σ. That is, $(s_1, \sigma, s_2), (s_1, \sigma, s_3) \in T$ implies $s_2 = s_3$. It is possible that transitions will go back to the same station; that is, $(s_1, \sigma, s_1) \in T$ is allowed.

Given a factory design, determine if the factory is capable of producing two distinct but indistinguishable products. If there is such a pair of products, then report the sum of the lengths of the strings in the pair that minimizes the sum of the lengths. If there is no such pair, print -1.

Input

The first line of input contains three integers, s ($1 \le s \le 50$), p ($1 \le p \le s$) and t ($1 \le t \le 52 \cdot s$), where s is the number of stations, p is the number of packaging stations, and t is the number of transitions. The set of stations is numbered from 1 to s.

Each of the next p lines contains a single integer i ($1 \le i \le s$). These are the packing stations. All values of i are distinct.

Each of the next t lines is of the form:

s_1 σ s_2

where s_1 and s_2 ($1 \le s_1, s_2 \le s$) are stations, and σ is a single character, consisting of an uppercase or lowercase letter. These are the transitions. The initial station is always station 1. There will never be two transitions out of the same state labelled with the same character.

Output

Output a single line with a single integer. If there is no pair of distinct, indistinguishable products, then output -1. If there exists a pair of distinct, indistinguishable products, then output the smallest sum $|a| + |b|$ where a and b are strings corresponding to distinct, indistinguishable products.

Sample Input 1

```
4 1 8
3
1 F 1
1 A 2
2 b 1
2 F 3
2 d 3
3 B 3
3 y 4
4 d 1
```

Sample Output 1

```
10
```

Sample Input 2

```
4 1 8
3
1 F 1
1 A 2
2 b 1
2 F 3
2 d 3
3 B 3
3 y 4
4 d 4
```

Sample Output 2

```
-1
```

Sample Input 3

```
5 2 5
1
5
1 i 2
2 c 3
3 p 4
4 c 1
1 I 5
```

Sample Output 3

```
4
```

This page is intentionally left blank.

Problem L
Triangular Logs
Time Limit: 12 Second(s)

The local forest has a lot of trees! Each tree is located at integer coordinates and has an integer height. Cutting down any tree gives you a log with a length equal to its height. You want to obtain three triangular logs (that is, three logs that form a non-degenerate triangle) by cutting down three trees.

Given a list of queries which each specify an axis-aligned rectangular region, can you obtain three triangular logs by cutting down three trees in that region, possibly including those on the boundary of the rectangle?

Input

The first line of input contains two integers n and q ($1 \leq n, q \leq 10^5$), where n is the number of trees and q is the number of queries.

Each of the next n lines contains three integers x, y and h ($1 \leq x, y, h \leq 10^9$), which describes a tree at location (x, y) with height h. All tree locations are distinct.

Each of the next q lines contains four integers x_{low}, y_{low}, x_{high} and y_{high} ($1 \leq x_{low} \leq x_{high} \leq 10^9$, $1 \leq y_{low} \leq y_{high} \leq 10^9$), describing an axis-aligned rectangular region for a query.

Output

Output q lines. Each line contains a single integer, which is the answer to the given query. Output `1` if there are three trees in the queried region that can form a non-degenerate triangle, and `0` otherwise. Output answers to the queries in the order of the input.

Sample Input 1

```
9 5
1 3 3
2 3 1
3 3 4
1 2 1
2 2 5
3 2 9
1 1 2
2 1 6
3 1 5
1 1 1 2
1 1 2 2
1 1 1 3
1 2 3 2
1 1 3 3
```

Sample Output 1

```
0
1
0
0
1
```

Problem M
Word Ladder
Time Limit: 1 Second(s)

You and Alice are solving word ladder puzzles. A *word ladder* is a sequence of words, all with the same number of letters, where each word in the sequence differs from the previous word by a single letter.

Because you are better at creating word ladders than at solving them, you have decided to devise a very specific puzzle for Alice. You will give Alice a list of n words. Then, she will construct a word ladder from your word list. Alice must start from the first word in your list, and modify one letter at a time to obtain the last word in your list. Each intermediate word Alice uses must also be from your word list.

Alice is so good at solving word ladder puzzles that she always produces the shortest word ladder possible. You want to force Alice to use all n words in her word ladder. There should be no way to construct a shorter word ladder from the starting word to the ending word using the words in your word list.

Create a list of n words, such that the shortest word ladder from the first word to the last word uses all the words in the list. Because you need to verify the word ladder solution before giving the puzzle to Alice, the word list should be given in word ladder order.

Input

The single line of input contains a single integer n ($3 \le n \le 5{,}000$), which is the length of the word list (and solution word ladder) you must construct for Alice.

Output

Output n lines. Each line contains a single word of length at most 10 letters. The words must all be distinct, of the same length, and consist only of lowercase letters. Note that for this problem, a word is simply a string of lowercase letters; it does not have to be an English word.

The list of n words should be printed in word ladder order, such that each word differs from the previous word by a single letter. There should exist no shorter word ladder from the first word to the last word using fewer than n words from the list.

It can be proven that an answer exists for all n satisfying the input constraint. Any answer satisfying these requirements will be considered correct.

Sample Input 1

```
5
```

Sample Output 1

```
lead
load
toad
told
gold
```

Sample Input 2

```
3
```

Sample Output 2

```
aa
ab
bb
```