Problem A
Edit Distance

A binary string is a non-empty sequence of 0's and 1's, e.g., `010110`, `1`, `11101`, etc. The edit distance of two binary strings S and T, denoted by $edit(S, T)$, is the minimum number of single-character edit (insert, delete, or substitute) to modify S into T. For example, the edit distance of `0011` and `1100` is 4, i.e. `0011` $\rightarrow$ `011` $\rightarrow$ `11` $\rightarrow$ `110` $\rightarrow$ `1100`. The edit distance of `1100101` and `1110100` is 2, i.e. `1100101` $\rightarrow$ `1110101` $\rightarrow$ `1110100`.

Ayu has a binary string S. She wants to find a binary string with the same length as S that maximizes the edit distance with S. Formally, she wants to find a binary string T_{max} such that $|S| = |T_{max}|$ and $edit(S, T_{max}) \geq edit(S, T')$ for all binary string T' satisfying $|S| = |T'|$.

She needs your help! However, since she wants to make your task easier, you are allowed to return any binary string T with the same length as S such that the edit distance of S and T is more than half the length of S. Formally, you must return a binary string T such that $|S| = |T|$ and $edit(S, T) > \frac{|S|}{2}$.

Of course, you can still return T_{max} if you want, since it can be proven that $edit(S, T_{max}) > \frac{|S|}{2}$ for any binary string S. This also proves that there exists a solution for any binary string S. If there is more than one valid solution, you can output any of them.

Input

Input contains a binary string S ($1 \leq |S| \leq 2000$).

Output

Output in a line a binary string T with the same length as S that satisfies $edit(S, T) > \frac{|S|}{2}$.

Sample Input #1

```
0011
```

Sample Output #1

```
1100
```

Explanation for the sample input/output #1

As illustrated in the example above, the edit distance of `0011` and `1100` is 4. Since $4 > \frac{4}{2}$, `1100` is one of the valid output for this sample.

Sample Input #2

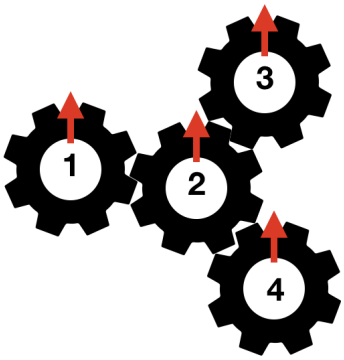
```
1100101
```

Sample Output #2

```
0011010
```

Problem B
Rotating Gears

Andi has N gears (numbered from 1 to N) on a board. Each gear has an arrow initially pointing upward. Some pair of gears may touch each other. If we construct a graph where each gear is a node, and we connect each pair of touching gears with an edge, then the structure of this graph is a tree. For example, the following is a possible gear configuration example with $N = 4$ gears and gear 2 touches all other gears.



Standard gear rotation rule applies: Suppose a gear u touches another gear v, and gear u is rotated α degrees clockwise, then gear v will be rotated α degrees counter-clockwise, and vice-versa.

Andi wants to perform three kinds of operations:

1. Take out a gear from its position on the board.
2. Place a gear back to its original position on the board. When doing this, Andi places a gear back in a way such that the arrow points to the same degree as when it was removed. Assume that it is always possible to do that, i.e. you do not need to concern with the mechanical aspect of the gear.
3. Rotate a gear α degrees clockwise.

Let δ_u be the arrow's degree (clockwise, modulo 360) of gear u after Q operations are done. Andi wants to know the sum of δ_u for all u. Furthermore, since rotating gears requires a lot of work, Andi also wants to know how much energy he needs for every Type 3 operation (rotating a gear). The amount of energy Andi needs to perform the Type 3 operation is defined as $\langle$number of rotating gears$\rangle \times \langle$rotation done in degrees$\rangle$

Input

Input begins with a line containing an integer N ($1 \leq N \leq 100000$) representing the number of gears. The next $N - 1$ lines, each contains two integers: $u\ v$ ($1 \leq u, v \leq N$; $u \neq v$) representing that gear u and gear v touches each other. It is guaranteed that the structure of the connected gears forms a tree. The next line contains an integer Q ($1 \leq Q \leq 100000$) representing the number of operations. The next Q lines, each representing an operation to be done sequentially. Each operation is given in one of the following formats:

- Two integers: $1\ x$ ($1 \leq x \leq N$). This means that this operation takes out gear x from its position on the board. It is guaranteed that gear x is currently on the board.
- Two integers: $2\ x$ ($1 \leq x \leq N$). This means that this operation places gear x back to its original position on the board. It is guaranteed that gear x is currently not on the board.
- Three integers: $3\ x\ \alpha$ ($1 \leq x \leq N$; $0 \leq \alpha \leq 359$). This means that this operation rotates gear x α degrees clockwise. It is guaranteed that gear x is currently on the board.

Output

For each Type 3 operation in the same order as input, output in a line the amount of energy needed to rotate the gears. Finally, output in a line the sum of δ_u for all u.

Sample Input	Sample Output
4 1 2 2 3 2 4 8 3 2 160 1 2 3 1 10 3 3 10 3 4 10 2 2 1 1 3 3 15	640 10 10 10 45 805

Explanation for the sample input/output

The structure of the gears for this sample is illustrated by the figure in the problem description. The following table illustrates the state of each gear after each operation.

After operation	δ_u			
	Gear 1	Gear 2	Gear 3	Gear 4
Initially	0	0	0	0
1	**200**	**160**	**200**	**200**
2	200	REMOVED	200	200
3	**210**	REMOVED	200	200
4	210	REMOVED	**210**	200
5	210	REMOVED	210	**210**
6	210	160	210	210
7	REMOVED	160	210	210
8	REMOVED	**145**	**225**	**225**

Therefore, the sum of δ_u for all u is $210 + 145 + 225 + 225 = 805$.

Problem C
Smart Thief

Ayu managed to steal Budi's treasure box and ready to uncover any secret Budi hides. Unfortunately, the treasure box has some security system to prevent any unauthorized person (e.g., Ayu) from opening it.

To unlock the treasure box, Ayu has to input a correct PIN (Personal Identification Number) of length N, which of course, she doesn't have. Ayu has no choice other than trying all possible PIN combinations. However, Ayu notices that this security system has an interesting (old) mechanism.

When you enter an N digits PIN, it is evaluated automatically and promptly, i.e. you don't need to push some "enter" button to confirm the PIN. Whenever your entered PIN is wrong, it removes the oldest (first) digit and shifts all the remaining to the left, thus, you only need to enter one more (last) digit to make it N length again.

For example, let $N = 4$. If we input `204320435`, then we actually test 6 PINs (with 5 different PINs):

- `[2043]20435`, tested PIN = `2043`
- `2[0432]0435`, tested PIN = `0432`
- `20[4320]435`, tested PIN = `4320`
- `204[3204]35`, tested PIN = `3204`
- `2043[2043]5`, tested PIN = `2043`
- `20432[0435]`, tested PIN = `0435`

Notice that `2043` is tested twice in this example.

As a CS student, Ayu knows that finding the correct PIN by trying all possible combinations can be very time-consuming, but alas, there's no other way. Ayu decides that she wants to test at least K different PINs on the first day. Your task is to help Ayu by simply giving her the string S which contains at least K different PINs. Ayu doesn't care which PIN she's going to test (so long at least there are K different PINs) or whether any PIN is tested more than once in S, but string S needs to be as short as possible. If there is more than one possible string for S, you can output any of them.

Note that, as this system is quite old, there are only M available digits ranging from 0 to 9.

Input

Input begins with a line containing three integers: N M K ($1 \leq N \leq 100000$, $1 \leq M \leq 10$, $1 \leq K \leq \min(M^N, 100000)$), representing the PIN length, the number of available digits, and the minimum number of PINs to be tested, respectively. The next line contains M integers: A_i ($0 \leq A_i \leq 9$), representing the available digits. You may assume all A_i are distinct. You may also assume that the values of N, M, and K are chosen such that the answer contains no longer than 100000 digits

Output

Output in a line the **shortest** string which contains at least K different PINs as its substring. If there is more than one such string, you can output any of them.

Sample Input #1

```
3 2 5
4 7
```

Sample Output #1

```
7477447
```

Explanation for the sample input/output #1

There are 5 different PINs of length 3 tested with the string `7477447`, i.e. `447`, `477`, `744`, `747`, and `774`.

Sample Input #2

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

Sample Output #2

```
1234554321
```

Explanation for the sample input/output #2

There are 9 different PINs of length 2 tested with the string `1234554321`, i.e. `12`, `21`, `23`, `32`, `34`, `43`, `45`, `54`, and `55`.

Sample Input #3

```
6 3 2
9 3 5
```

Sample Output #3

```
9353593
```

Explanation for the sample input/output #3

There are 2 different PINs of length 6 tested with the string `9353593`, i.e. `353593`, and `935359`.

Problem D
Icy Land

You are given a board of R rows and C columns where each cell is either a dry land ('#') or an icy land ('.'). You can move to either one of the four directions (north, south, east, or west) in the board.

You can walk or standstill on any dry land without any problem; however, an icy land is very slippery. If you stand at cell (r, c) and decide to move on a particular direction, then you will move on that direction continuously until you reach a dry land or the border of the board.

For example, consider the following board of 6 rows and 7 column.

```
#.....
.#....
..##...
..##...
.......
.#.....
```

- From $(1, 3)$, the 1^{st} row and 3^{rd} column, if you move to the south, then you will end up at $(3, 3)$.
- From $(4, 4)$, if you move to the north, then you will end up at $(3, 4)$.
- From $(1, 1)$, if you move to the east, then you will end up at $(1, 7)$.
- From $(6, 5)$, if you move to the west, then you will end up at $(6, 2)$.

Supposed your initial position is at cell $(3, 7)$ and your moves are west, west, west, south, east, north, west, and west, respectively. Then your positions are $(3, 7) \rightarrow (3, 4) \rightarrow (3, 3) \rightarrow (3, 1) \rightarrow (6, 1) \rightarrow (6, 2) \rightarrow (2, 2) \rightarrow (2, 1) \rightarrow (2, 1)$, thus, visiting 8 different cells. Note that cell $(2, 1)$ is visited twice. Cells in which you only passed through are also not considered as visited, e.g., in the example above, if you move south from $(3, 1)$, then you will pass through $(4, 1)$ and $(5, 1)$, and arrive at $(6, 1)$; thus, $(4, 1)$ and $(5, 1)$ are not considered as visited from that particular move, only $(3, 1)$ and $(6, 1)$ are.

You are allowed to change any icy land into a dry land, and your goal is to make sure that you can always visit **all** the cells in the board **regardless** of your initial starting position; in other words, you do not know your starting position yet, but given the board, you want to make sure that you can achieve your goal. What is the minimum number of cells you need to change to ensure that your goal can be achieved?

Input

Input begins with a line containing two integers: $R\ C$ ($1 \leq R, C \leq 500$) representing the number of rows and columns of the board, respectively. The next R lines, each contains C characters representing the given board. Each character is either '#' which represents a dry land or '.' which represents an icy land.

Output

Output contains an integer in a line representing the minimum number of cells you need to change to ensure that you can visit all the cells in the board regardless of your initial starting position.

Sample Input

```
4 4
....
.###
##..
###.
```

Sample Output

```
1
```

Explanation for the sample input/output

We only need to change cell $(3, 3)$.

```
....
.###
###.
###.
```

Let 'N' be north, 'S' be south, 'W' be west, and 'E' be east.

- If you start at cell $(1, 1)$, then the movement "SSENNWENSESSEWNENWNE" will visit all the cells. The corresponding positions are: $(1, 1) \rightarrow (3, 1) \rightarrow (4, 1) \rightarrow (4, 2) \rightarrow (3, 2) \rightarrow (2, 2) \rightarrow (2, 1) \rightarrow (2, 2) \rightarrow (1, 2) \rightarrow (2, 2) \rightarrow (2, 3) \rightarrow (3, 3) \rightarrow (4, 3) \rightarrow (4, 4) \rightarrow (4, 3) \rightarrow (3, 3) \rightarrow (3, 4) \rightarrow (2, 4) \rightarrow (2, 3) \rightarrow (1, 3) \rightarrow (1, 4)$.
- If you start at cell $(1, 2)$, then the movement "W" followed by the movement in the first example above ("SSENNWENSESSEWNENWNE") will visit all the cells. The corresponding positions are: $(1, 2) \rightarrow (1, 1) \rightarrow \ldots$
- If you start at cell $(1, 3)$, then the movement "W" followed by the movement in the first example above will visit all the cells. The corresponding positions are: $(1, 3) \rightarrow (1, 1) \rightarrow \ldots$
- $\ldots$
- If you start at cell $(4, 3)$, then the movement "NNNW" followed by the movement in the first example above will visit all the cells. The corresponding positions are: $(4, 3) \rightarrow (3, 3) \rightarrow (2, 3) \rightarrow (1, 3) \rightarrow (1, 1) \rightarrow \ldots$
- If you start at cell $(4, 4)$, then the movement "NNW" followed by the movement in the first example above will visit all the cells. The corresponding positions are: $(4, 4) \rightarrow (2, 4) \rightarrow (1, 4) \rightarrow (1, 1) \rightarrow \ldots$

Problem E
Artilleries and Defensive Walls

Sauville has been engulfed in a devastating war with its neighboring country, Norland, for centuries. Their border can be represented as a straight horizontal line $y = 0$, where all areas for $y < 0$ belong to Sauville and all areas for $y > 0$ belong to Norland.

Norland has deployed N of its artilleries at position (x_i, y_i) where $y_i > 0$. In addition, Norland also has built M defensive walls. Each defensive wall can be represented as a tuple $\langle x_{j1}, x_{j2}, y_j \rangle$, which is a horizontal line segment spanned from (x_{j1}, y_j) to (x_{j2}, y_j), where $x_{j1} < x_{j2}$ and $y_j > 0$. It is known to Sauville that no Norland's artillery is located at any of its defensive walls (including its endpoints), and no two artilleries are at the same position. It is also known that no two walls (including their endpoints) are sharing a common point.

Sauville has decided to build a watchtower to observe any suspicious activity on any Norland's artilleries. As the cost to build one watchtower is almost astronomical for Sauville, they can only afford to build one. Thus, Q position candidates (x_k, y_k) where $y_k < 0$ for the watchtower have been proposed. If the watchtower is built at (x, y), then all artilleries which lie on the *line-of-sight* from (x, y) can be observed (visible) by the watchtower. A position (x_i, y_i) lies on the line-of-sight from (x, y) if and only if the straight line connecting (x_i, y_i) and (x, y) does not intersect with any defensive walls (including its endpoints); in other words, there should be no point (x', y') such that (x', y') lies on a defensive wall and also on the line segment between (x_i, y_i) and (x, y). Note that an artillery does not affect the visibility of any other artilleries from the watchtower.

Your task in this problem is to determine the number of Norland's artilleries which can be observed from each proposed watchtower position.

Input

Input begins with a line containing three integers: N M Q ($1 \le N \le 40000$; $0 \le M \le 5$; $1 \le Q \le 40000$) representing the number of artilleries, the number of defensive walls, and the number of proposed watchtower positions, respectively. The next N lines, each contains two integers: x_i y_i ($-10^6 \le x_i \le 10^6$; $0 < y_i \le 10^6$) representing the position of the i^{th} artillery for $i = 1 \ldots N$. The next M lines, each contains three integers: x_{j1} x_{j2} y_j ($-10^6 \le x_{j1} < x_{j2} \le 10^6$; $0 < y_j \le 10^6$) representing the position of the j^{th} defensive wall for $j = 1 \ldots M$. The next Q lines, each contains two integers: x_k y_k ($-10^6 \le x_k \le 10^6$; $-10^6 \le y_k < 0$) representing a proposed position for the watchtower.

Output

For each proposed watchtower position in the same order as input, output in a line a single integer representing the number of Norland's artilleries which can be observed from the proposed position.

Sample Input

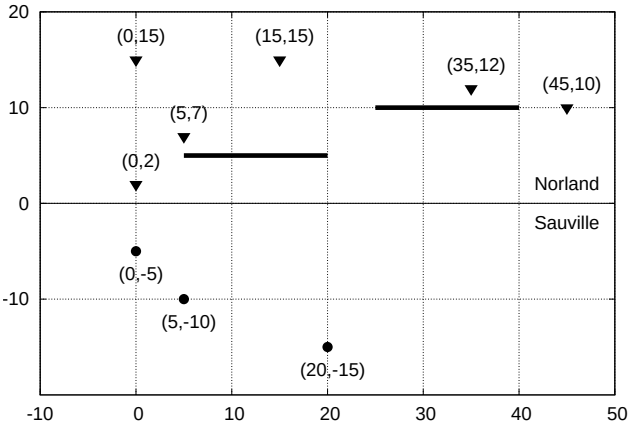
```
6 2 3
0 2
0 15
5 7
15 15
35 12
45 10
5 20 5
25 40 10
0 -5
5 -10
20 -15
```

Sample Output

```
4
3
2
```

Explanation for the sample input/output

The position of all artilleries, defensive walls, and proposed watchtowers are shown in the following figure.



- The first proposed watchtower $(0, -5)$ can observe 4 artilleries, i.e. at $(0, 2)$, $(0, 15)$, $(5, 7)$, and $(45, 10)$.
- The second proposed watchtower $(5, -10)$ can observe 3 artilleries, i.e. at $(0, 2)$, $(0, 15)$, and $(45, 10)$.
- The third proposed watchtower $(20, -15)$ can observe 2 artilleries, i.e. at $(0, 2)$ and $(45, 10)$.

Problem F
Popping Balloons

Ayu is participating in ABC 2018 (Arranging Blocks Competition). In this competition, each contestant is given M minutes and N tasks which should be solved one-by-one in the given order. The contestant who solves the most tasks is the winner. What makes this contest interesting to you (ICPC contestants) is that this contest uses a similar encouragement as ICPC, i.e. balloons. In particular, each time a contestant solves a task, s/he will be given a balloon.

Ayu is convinced that she can defeat all other contestants, except one particular contestant, Budi, her rival. Ayu knows well her skill, i.e. she knows exactly how long it takes for her to solve a particular task. Unsurprisingly, Ayu also knows well Budi's skill (they are rival for a reason). Let there be two arrays of integers $A_{1..N}$ and $B_{1..N}$. A_i denotes the time (in minutes) needed by Ayu to solve the i^{th} task, while B_i denotes the time (in minutes) needed by Budi to solve the same i^{th} task.

Here comes the interesting part. Ayu knows that Budi is very sensitive to any disturbingly loud sound like a balloon being popped. Whenever Budi is surprised (due to a balloon being popped), he will lose his concentration and has to repeat the task he's doing. For example, suppose Budi needs 10 minutes to solve a particular task. If a balloon pops at the 7^{th} minute, then Budi repeats the task at the 7^{th} minute (out of his 10 minutes), causing him to solve the task with $7+10=17$ minutes. If two balloon pops, each at the 7^{th} and 13^{th} minute, then Budi repeats the task at minute 7^{th} (out of his 10 minutes), repeats it again at minute 6^{th} (out of his 10 minutes), and finally solved the task with $7+6+10=23$ minutes. If a balloon pops at the same time Budi supposed to solve the task (i.e. at the 10^{th} minute in this example), then Budi will also not solve that task. Therefore, Budi has to spend another 10 minutes (for a total of $10+10=20$ minutes) to solve that particular task in this case.

Ayu plans to exploit Budi's weakness in order to defeat him, i.e. Ayu will strategically use the balloons (popping them at integer minutes) she gets from solving the tasks. Your task in this problem is to find out whether it is possible for Ayu to have the number of solved tasks to be strictly larger than Budi's. If it is possible, you should output one (any) working plan on when she should pop the balloons.

Note that if Ayu solves a task at exactly the M^{th} minute, then the task is considered as solved. Similarly, if Budi solves a task at exactly the M^{th} minute, then the task is considered as solved, except if Ayu decides to pop a balloon at the same time. Also, Ayu can pop a balloon as soon as she receives it. Ayu cannot pop more than one balloon at the same minute. She also cannot pop any balloon after the M^{th} minute mark.

Input

Input begins with a line containing two integers: N M ($1 \leq N \leq 100000$; $1 \leq M \leq 10^9$) representing the number of tasks and duration of the competition, respectively. The second line contains N integers A_i ($1 \leq A_i \leq 10^9$) representing the time needed by Ayu to solve the i^{th} task. The third line contains N integers B_i ($1 \leq B_i \leq 10^9$) representing the time needed by Budi to solve the i^{th} task.

Output

If it is not possible for Ayu to win the competition by having **strictly** larger number of solved tasks than Budi, simply output -1 in a line. Otherwise, output begins with an integer K in a line representing the number of balloons Ayu needs to pop. The next line contains K integers (each separated by a single space), sorted by **increasing order**, representing the minute in which Ayu should pop a balloon. You may output any configuration as long as it's valid, i.e. Ayu has at least one balloon **when** she pops a balloon, Ayu is not popping a balloon after the contest is over, Ayu is not popping more than one balloon at the same minute, and the configuration causes Ayu to have a larger number of solved tasks than Budi.

Sample Input #1	Sample Output #1
4 30 9 10 10 10 4 10 5 10	2 12 19

Explanation for the sample input/output #1

Ayu gets her first balloon at minute 9, while at that time, Budi already solved the first task (at minute 4) and is currently doing his second task which needs 5 more minutes (projected to finish at minute 14). Ayu pops his first balloon at minute 12, i.e. 2 minutes before Budi finish the second task, causing Budi to repeat the second task. Now, the projected time for Budi to finish the second task is at minute 22. At minute 19, Ayu gets her second balloon and pops it right away, causing Budi to repeat the second task again. Now, the projected time for Budi to finish the second task is at minute 29. At minute 29, Ayu gets his third balloon, and at the same time, Budi also solved the second task. The competition ends at minute 30. In total, Ayu solved 3 tasks, while Budi only managed to solve 2 tasks.

Sample Input #2	Sample Output #2
5 50 10 10 10 10 10 15 12 19 17 20	0

Sample Input #3	Sample Output #3
5 10 15 10 5 5 5 9 10 10 10 10	-1

Explanation for the sample input/output #3

Ayu cannot solve the first task during the competition, so no balloon for her to play with.

Problem G

Go Make It Complete

Andi has a network G with N machines and M links; each link connects exactly two different machines. Due to some circumstances, Andi has to make his network "complete", i.e. every machine is directly linked to every other machine. This means Andy has to add a new link between any pair of machines which are not already directly linked.

In order to accomplish his goal, Andi outsources the work to a company, Go. Go accepts any work order on a network G with a requested integer k, i.e. $go(G, k)$. The way Go works: First, it creates a list L containing **all** pair of machines which are **not** yet directly linked. Then, Go evaluates each pair of machines (a, b) from L and create a new link between them if $\delta_a + \delta_b \geq k$. Note that δ_a is the degree of machine a, i.e. the number of links a has at the time of evaluation. Similarly, δ_b is for machine b.

The problem with Go's procedure is it evaluates each pair of machines in the order appeared in L. For example, let G be a network with $N = 4$ machines (machine 1, 2, 3, and 4) and $M = 3$ links; the links are $\{(1, 2), (2, 3), (3, 4)\}$. The degree of each machine before a work order is requested is: $\delta_1 = 1$, $\delta_2 = 2$, $\delta_3 = 2$, $\delta_4 = 1$, or simply can be written as $[1, 2, 2, 1]$. Let say a work order with $k = 3$ is requested ($go(G, 3)$).

Consider these two lists:

- $L_1 = ((1, 4), (1, 3), (2, 4))$.
  - × Evaluate $(1, 4)$ and don't create a new link since $\delta_1 + \delta_4 = 1 + 1 = 2$. The degree is still $[1, 2, 2, 1]$.
  - ✓ Evaluate $(1, 3)$ and create a new link since $\delta_1 + \delta_3 = 1 + 2 = 3$. The degree becomes $[2, 2, 3, 1]$.
  - ✓ Evaluate $(2, 4)$ and create a new link since $\delta_2 + \delta_4 = 2 + 1 = 3$. The degree becomes $[2, 3, 3, 2]$.

  The final result is a network with 5 links, which is not complete as it misses the $(1, 4)$ link.
- $L_2 = ((2, 4), (1, 3), (1, 4))$.
  - ✓ Evaluate $(2, 4)$ and create a new link since $\delta_2 + \delta_4 = 2 + 1 = 3$. The degree becomes $[1, 3, 2, 2]$.
  - ✓ Evaluate $(1, 3)$ and create a new link since $\delta_1 + \delta_3 = 1 + 2 = 3$. The degree becomes $[2, 3, 3, 2]$.
  - ✓ Evaluate $(1, 4)$ and create a new link since $\delta_1 + \delta_4 = 2 + 2 = 4$. The degree becomes $[3, 3, 3, 3]$.

  The final result is a network with 6 links and complete.

As we can see, L_2 produces a complete network, while L_1 is not.

Ordering a work to Go can be very expensive with lower k, thus, Andi has to make k as high as possible while maintaining the possibility that a complete network can be achieved with k. In other words, Andi wants the highest k such that there exists L such that $go(G, k)$ can produce a complete network, and for any j ($j > k$), there is no valid L for $go(G, j)$ which can produce a complete network. Your task in this problem is to find such k.

Input

Input begins with a line containing two integers: N M ($2 \leq N \leq 500$; $0 \leq M < \frac{N \times (N-1)}{2}$) representing the number of machines and the number existing links, respectively. The machines are numbered from 1 to N. The next M lines, each contains two integers: a_i b_i ($1 \leq a_i < b_i \leq N$) representing an existing link connecting machine a_i and b_i. You are guaranteed that each pair (a_i, b_i) will appear at most once in the input.

Output

Output contains an integer k in a line, as requested.

Sample Input #1

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

Sample Output #1

```
3
```

Sample Input #2

```
5 0
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

Andi's network has no link at all at the beginning. To make his network complete, Andi has to order $go(G, 0)$.

Sample Input #3

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

Sample Output #3

```
2
```

Problem H
Lexical Sign Sequence

Andi likes numbers and sequences, especially, sign sequences. A sign sequence is a sequence which consists of -1 and 1. Andi is a curious person, thus, he wants to build a sign sequence which length is N (the positions are numbered from 1 to N, inclusive).

However, Andi also likes some challenges. Therefore, he prefilled some positions in the sequence with -1 or 1 (the number in these positions cannot be changed). Andi also wants the sequence to fulfill K constraints. For each constraint, there are 3 numbers: A_i, B_i, and C_i. This means that the sum of numbers which position is in the range $[A_i, B_i]$ (inclusive) must be at least C_i.

Sounds confusing, right? It is not done yet. Since there can be many sequences that fulfill all the criteria above, Andi wants the sequence to be lexicographically smallest. Sequence X is lexicographically smaller than sequence Y if and only if there exists a position i where $X_i < Y_i$ and $X_j = Y_j$ for all $j < i$.

Find the sequence Andi wants.

Input

Input begins with a line containing two integers: N K ($1 \leq N \leq 100000$; $0 \leq K \leq 100000$) representing the length of the sequence and the number of constraints, respectively. The second line contains N integers: P_i ($-1 \leq P_i \leq 1$). If $P_i = 0$, then the i^{th} position in the sequence is not prefilled, otherwise the i^{th} position in the sequence is prefilled by P_i. The next K lines, each contains three integers: A_i B_i C_i ($1 \leq A_i \leq B_i \leq N$; $-N \leq C_i \leq N$) representing the i^{th} constraint.

Output

Output contains N integers (each separated by a single space) in a line representing the sequence that Andi wants if there exists such sequence, or "`Impossible`" (without quotes) otherwise.

Sample Input #1

```
3 2
0 0 0
1 2 2
2 3 -1
```

Sample Output #1

```
1 1 -1
```

Explanation for the sample input/output #1

Both sequences $[1, 1, -1]$ and $[1, 1, 1]$ satisfy the prefilled conditions and the given K constraints. The former is lexicographically smaller.

Sample Input #2

```
3 2
0 -1 0
1 2 2
2 3 -1
```

Sample Output #2

```
Impossible
```

Explanation for the sample input/output #2

The second position is already prefilled with -1, so it is impossible to fulfill the first constraint. There is no valid sequence in this case.

Problem I
Lie Detector

Andi is a young and prominent detective in the police force. His ability to track down criminals, uncover the truth, and solve cases never ceases to amaze all of his colleagues. One day, he is faced with a suspicious eyewitness testimony when working on a certain case. In usual cases, Andi simply ignores such unreliable testimony; however, in this case, the eyewitness testimony is too important to be ignored. To resolve this situation, Andi has to rely on technology, i.e. using a lie detector.

Andi proceeds to use a lie detector to detect whether the eyewitness testimony is true. However, Andi notices that the lie detector he used might have been tampered, thus, he employs a second lie detector to detect whether the first lie detector's result is correct. This situation happens repeatedly such that Andi ends up employing N lie detectors in total. The i^{th} lie detector reports the truth of the $(i-1)^{th}$ lie detector for $i = 2..N$, and the 1^{st} lie detector reports the truth of the eyewitness testimony.

In the end, Andi knows that the last (N^{th}) lie detector has not been tampered and always report the truth correctly. Now, he needs to determine whether the eyewitness testimony is true given the result of all lie detectors.

For example, let $N = 4$ and the lie detectors result are $(\texttt{LIE}, \texttt{LIE}, \texttt{TRUTH}, \texttt{TRUTH})$.

- The 4^{th} lie detector reports that the 3^{rd} lie detector is `TRUTH`. As the 4^{th} lie detector always report the truth correctly, then the 3^{rd} lie detector's result is correct as it is.
- The 3^{rd} lie detector reports that the 2^{nd} lie detector is `TRUTH`. As the 3^{rd} lie detector's result is correct as it is, then the 2^{nd} lie detector's result is also correct as it is.
- The 2^{nd} lie detector reports that the 1^{st} lie detector is `LIE`. As the 2^{nd} lie detector's result is correct as it is, then the 1^{st} lie detector's result is wrong.
- The 1^{st} lie detector reports that the eyewitness testimony is `LIE`. As the 1^{st} lie detector's result is wrong, then the eyewitness testimony is correct; in other words, what the eyewitness says is true.

Therefore, the eyewitness testimony in this example is true.

Input

Input begins with a line containing an integer N ($2 \leq N \leq 100000$). The next N lines, each contains a string S_i (either `TRUTH` or `LIE`) representing the output of the i^{th} lie detector for $i = 1..N$ respectively.

Output

Output contains a string `TRUTH` or `LIE` in a line whether the eyewitness testimony is true or false.

Sample Input #1

```
4
LIE
LIE
TRUTH
TRUTH
```

Sample Output #1

```
TRUTH
```

Explanation for the sample input/output #1

This sample is illustrated in the problem description above.

Sample Input #2

```
3
LIE
LIE
LIE
```

Sample Output #2

```
LIE
```

Problem J
Future Generation

Andi is getting married! He and his partner plan to have N children. To avoid any hassle in the future, Andi wants to decide all their children's name in advance. Specifically, he wants each child to have a name which is lexicographically **larger** than any of his/her older siblings. Of course, his partner also agrees with this idea. String A is lexicographically larger than string B if and only if B is a prefix of A or there exists an index i where $A_i > B_i$ and $A_j = B_j$ for all $j < i$. Note that a proper name can be as short as one character, but it **cannot** be an empty string.

Life is good for Andi, until one day he told his soon-to-be-grandmother-in-law (i.e. his partner's grandma) about this marriage plan. After learning that Andi plans to have N children with her granddaughter, she gave him N names to be used. Moreover, the i^{th} name can only be used for the i^{th} child.

After going through a long debate with her grandma, Andi came into an agreement: The i^{th} child's name should be a subsequence of the i^{th} name her grandma gave. A string A is a subsequence of string B if A can be obtained by deleting zero or more characters from B without changing the remaining characters' order, e.g., `ABC` is a subsequence of `DAEBCCB`, but `EFG` is not a subsequence of `FABEGC`.

Even though Andi dislikes the given list of names, he still wants to impress his partner by showing that he can satisfy both her grandma's wish and his own desire (i.e. each child's name is lexicographically larger than any of his/her older siblings). However, Andi wonders, what is the maximum possible total length of their children names?

For example, let $N = 3$, and the names given by his partner's grandma are (`KARIM`, `PARBUDI`, `CHANDRA`). Here are several example set of names which satisfies Andi's desire:

- [`AR`, `BI`, `CRA`] with a total length of $2 + 2 + 3 = 7$.
- [`ARI`, `BUDI`, `CHANDRA`] with a total length of $3 + 4 + 7 = 14$.
- [`ARIM`, `ARUDI`, `CHANDRA`] with a total length of $4 + 5 + 7 = 16$.
- [`AIM`, `ARBUDI`, `CHANDRA`] with a total length of $3 + 6 + 7 = 16$.
- $\cdots$

Among all sets of names which satisfy Andi's desire in this example, the maximum total length is 16. Note that there might be cases where valid set of names cannot be obtained. In such case, you should output `-1`. For example, let $N = 2$ and the names given by his partner's grandma are (`ZORO`, `ANDI`). In this example, all subsequences of the 2^{nd} name are lexicographically smaller than all subsequences of the 1^{st} name, thus, no possible valid set of names can be obtained.

Input

Input begins with a line containing an integer N ($1 \leq N \leq 15$) representing the number of children. The next N lines, each contains a string S_i ($1 \leq |S_i| \leq 15$) representing the i^{th} name given by Andi's soon-to-be-grandmother-in-law. S_i contains only uppercase alphabets ($S_{ij} \in \{A - Z\}$).

Output

Output contains an integer in a line representing the maximum possible total length of their children names, or `-1` if no possible valid set of names can be obtained.

Sample Input #1

```
3
KARIM
PARBUDI
CHANDRA
```

Sample Output #1

```
16
```

Sample Input #2

```
2
ZORO
ANDI
```

Sample Output #2

```
-1
```

Sample Input #3

```
7
HAVE
FUN
IN
ICPC
JAKARTA
TWENTY
EIGHTEEN
```

Sample Output #3

```
25
```

Explanation for the sample input/output #3

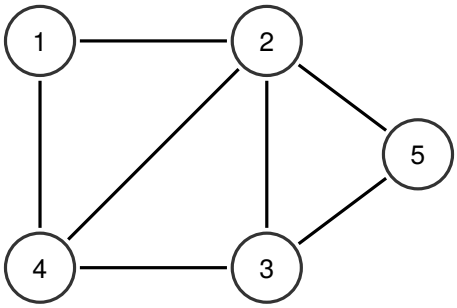
One possible solution is [AVE, FUN, IN, IPC, JAKARTA, NTY, TEEN] with a total length of $3+3+2+3+7+3+4 = 25$.

Problem K
Boomerangs

Let $G = (V, E)$ be a simple undirected graph with N vertices and M edges, where $V = \{1, \ldots, N\}$. A tuple $\langle u, v, w \rangle$ is called as boomerang in G if and only if $\{(u, v), (v, w)\} \subseteq E$ and $u \neq w$; in other words, a boomerang consists of two edges which share a common vertex.

Given G, your task is to find as many disjoint boomerangs as possible in G. A set S contains disjoint boomerangs if and only if each edge in G only appears at most once (in one boomerang) in S. You may output any valid disjoint boomerangs, but the number of disjoint boomerangs should be maximum.

For example, consider a graph $G = (V, E)$ of $N = 5$ vertices and $M = 7$ edges where $E = \{(1, 2), (1, 4), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5)\}$.



The maximum number of disjoint boomerangs in this example graph is 3. One example set containing the 3 disjoint boomerangs is $\{\langle 4, 1, 2 \rangle, \langle 4, 3, 2 \rangle, \langle 2, 5, 3 \rangle\}$; no set can contain more than 3 disjoint boomerangs in this example.

Input

Input begins with a line containing two integers: N M ($1 \leq N, M \leq 100000$), representing the number of vertices and the number edges in G, respectively. The next M lines, each contains two integers: u_i v_i ($1 \leq u_i < v_i \leq N$), representing the edge (u_i, v_i) in G. You may safely assume that each edge appears at most once in the given list.

Output

The first line of output contains an integer: K, representing the maximum number of disjoint boomerangs in G. The next K lines, each contains three integers: u v w (each separated by a single space), representing a boomerang $\langle u, v, w \rangle$. All boomerangs in the output should be disjoint. If there is more than one valid solution, you can output any of them.

Sample Input #1

```
5 7
1 2
1 4
2 3
2 4
2 5
3 4
3 5
```

Sample Output #1

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

Sample Input #2

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

Sample Output #2

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

Sample Input #3

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

Sample Output #3

```
1
2 1 3
```

Problem L
Binary String

A binary string is a non-empty sequence of 0's and 1's, e.g., `010110`, `1`, `11101`, etc. Ayu has a favorite binary string S which contains no leading zeroes. She wants to convert S into its **decimal** representation with her calculator.

Unfortunately, her calculator cannot work on any integer larger than K and it will crash. Therefore, Ayu may need to remove zero or more bits from S while maintaining the order of the remaining bits such that its decimal representation is no larger than K. The resulting binary string also must not contain any leading zeroes.

Your task is to help Ayu to determine the minimum number of bits to be removed from S to satisfy Ayu's need.

For example, let S = `1100101` and $K = 13$. Note that `1100101` is 101 in decimal representation, thus, we need to remove several bits from S to make it no larger than K. We can remove the 3^{rd}, 5^{th}, and 6^{th} most significant bits, i.e. `1100101` $\rightarrow$ `1101`. The decimal representation of `1101` is 13, which is no larger than $K = 13$. In this example, we removed 3 bits, and this is the minimum possible (If we remove only 2 bits, then we will have a binary string of length 5 bits; notice that any binary string of length 5 bits has a value of at least 16 in decimal representation).

Input

Input begins with a line containing an integer K ($1 \leq K \leq 2^{60}$) representing the limit of Ayu's calculator. The second line contains a binary string S ($1 \leq |S| \leq 60$) representing Ayu's favorite binary string. You may safely assume S contains no leading zeroes.

Output

Output contains an integer in a line representing the minimum number of bits to be removed from S.

Sample Input #1

```
13
1100101
```

Sample Output #1

```
3
```

Explanation for the sample input/output #1

This sample is illustrated by the example given in the problem description above.

Sample Input #2

```
13
1111111
```

Sample Output #2

```
4
```

Explanation for the sample input/output #2

Ayu must remove 4 bits to get 111, which is 7 in its decimal representation.