Problem A. Belarusian State University

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Being a student of Belarusian State University (BSU) is an earnest reason for pride. While studying the Theory of Algorithms course, you are obliged to solve many challenging problems before you are admitted to the final exam. Here is one of these problems.

You are given a positive integer n and $4n$ integers $c(i, j, k)$ which can be equal to 0 or 1 ($0 \le i < n$, $j \in \{0, 1\}$, $k \in \{0, 1\}$).

Consider two integers x and y between 0 and $2^n - 1$, inclusively. Let $x = \sum_{i=0}^{n-1} x_i \cdot 2^i$ and $y = \sum_{i=0}^{n-1} y_i \cdot 2^i$ be their binary representations ($x_i, y_j \in \{0, 1\}$). Define $f(x, y) = \sum_{i=0}^{n-1} c(i, x_i, y_i) \cdot 2^i$. Clearly, $f(x, y)$ is also an integer between 0 and $2^n - 1$.

Given two multisets A and B, find the multiset of values $f(a, b)$ over all pairs (a, b), where $a \in A$, $b \in B$.

Input

The first line contains an integer n ($1 \le n \le 18$).

The second line contains n binary strings of 4 digits. The i-th string consists of the values of $c(i-1, 0, 0)$, $c(i-1, 0, 1)$, $c(i-1, 1, 0)$, $c(i-1, 1, 1)$ in this particular order.

The next two lines describe multisets A and B, respectively. The description of a multiset consists of 2^n integers $q_0, q_1, \ldots, q_{2^n-1}$ denoting the quantities of the numbers $0, 1, \ldots, 2^n - 1$ in the multiset ($q_i \ge 0$, $\sum q_i \le 10^9$). There are no other numbers in the multisets.

Output

Print 2^n integers in a single line, the quantities of the numbers $0, 1, \ldots, 2^n - 1$ in the resulting multiset.

Examples

standard input	standard output
3 0111 0110 0001 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 1
2 1100 1101 2 0 2 1 2 0 2 1	2 4 3 16
1 0000 142857142 857142857 998244353 1755646	999999998000000001 0

Note

In the first example, you are given 5 and 6. For $x_i, y_i \in \{0, 1\}$, we have

$$f(x_0 + 2x_1 + 4x_2,\ y_0 + 2y_1 + 4y_2) = (x_0 \text{ OR } y_0) + 2 \cdot (x_1 \text{ XOR } y_1) + 4 \cdot (x_2 \text{ AND } y_2).$$

Thus, the only number in the resulting multiset is 7.

Problem C. Brave Seekers of Unicorns

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are a member of the Brave Seekers of Unicorns (BSU), the secret magical order. The BSU is fond of seeking unicorns. Recently, they have agreed to call an array $a_1, a_2, \ldots, a_k$ of k integers a *unicorn* if it satisfies the following conditions:

- the array is not empty ($k > 0$);
- there are no three consecutive elements with their bitwise XOR equal to zero ($a_i \oplus a_{i+1} \oplus a_{i+2} \neq 0$ for all $1 \le i \le k-2$);
- the array is strictly increasing ($a_i < a_{i+1}$ for all $1 \le i \le k-1$);
- the elements of the array are integers between 1 to n, inclusively ($1 \le a_i \le n$ for all $1 \le i \le k$).

For example, if $n = 10$, then the array $[1, 4, 5, 9]$ is not a unicorn because $1 \oplus 4 \oplus 5 = 0$, but the array $[2, 4, 7, 9]$ is a unicorn.

The Grand Master of the BSU has commanded you to calculate the number of unicorns. Since the number can be pretty large, you must compute it modulo 998 244 353.

Input

The only line contains an integer n ($1 \le n \le 10^6$).

Output

Print the number of unicorns modulo 998 244 353.

Examples

standard input	standard output
1	1
2	3
3	6
5	26
322	782852421

Problem D. Bank Security Unification

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The Bytelandian government has issued the *Bank Security Unification* law (or, shortly, the BSU law). The recent law regulates the usage of Wi-Fi routers in banks and other financial institutions.

According to the BSU law, all the n Wi-Fi routers in a bank must be located in a straight line. Suppose that the i-th router operates at the frequency f_i. Denote the *security* of a connection between two adjacent routers as f_i & f_{i+1}, where & is the bitwise AND operation.

A set of at least two routers numbered $1 \le i_1 < i_2 < \cdots < i_k \le n$ must be chosen as *active*. All other routers will be kept inactive so that they can replace the active ones if any of them would break. Denote the *security of the network* as the sum of the securities of all connections between adjacent active routers. In other words, the security of the network is calculated as $\sum_{j=1}^{k-1} f_{i_j} \,\&\, f_{i_{j+1}}$.

You are an employee of a large Bytelandian bank. Surely, the bank is obliged to comply with the BSU law. The routers are already placed in a line, and their placement cannot be changed. Now you want to choose some of the routers as active to maximize the security of the network.

Input

The first line contains an integer n, denoting the number of Wi-Fi routers in the bank ($2 \le n \le 10^6$).

The second line contains n integers $f_1, f_2, \ldots, f_n$, where f_i is the frequency of the i-th router in the line ($0 \le f_i \le 10^{12}$).

Output

Print the maximum possible security of the network.

Examples

standard input	standard output
5 1 2 3 1 3	5
4 1 2 4 0	0

Problem E. Brief Statements Union

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Egor learned about the secret organization called *Brief Statements Union* (BSU), whose ultimate goal is to make statements of all competitive programming problems clear and concise, and eliminate those long, boring, and unnecessary tales in the statements.

Egor decided to join the organization. For this purpose, he wrote the following problem with a short statement:

You are given an integer n *and* k *conditions. The* i*-th condition states that bitwise AND of all integers* $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ *is equal to* x_i*.*

For each condition i*, determine if there exists an array* $a_1, a_2, \ldots, a_n$ *of* n *integers which satisfies all the conditions except the condition* i*. Note that it is OK if the array satisfies the condition* i *too.*

The committee of the organization liked Egor's problem statement. And this is how he got accepted into the organization. Now, Egor has decided to offer the problem to this contest, so you have to solve it.

Input

The first line contains two integers n and k, denoting the required length of the array and the number of the conditions ($1 \le n, k \le 10^6$).

Then k lines follow, the i-th of them contains three integers l_i, r_i, x_i, describing the i-th condition ($1 \le l_i \le r_i \le n$, $0 \le x_i \le 10^{18}$).

Output

Print the binary string of k characters. The i-th character must be equal to '1' if there is an array of length n which satisfies all the conditions with the i-th one being removed. Otherwise, the i-th character must be equal to '0'.

Examples

standard input	standard output
4 3 1 2 1 2 4 3 2 2 1	011
4 3 1 2 1 3 4 3 2 3 1	111

Problem G. Biological Software Utilities

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are developing a software kit named *Biological Software Utilities* (BSU). The kit includes a program that is dedicated to tree recognition. Recall that a *tree* is a connected undirected graph without cycles.

In nature, when a tree grows, two neighboring vertices are added at the same time. Thus, you consider a tree to be *plausible* if, after removing some edges, the resulting graph consists only of connected components with 2 vertices. In other words, a tree is plausible if and only if it has a perfect matching.

Now you are to implement a new function for BSU to calculate the number of plausible trees that have n vertices numbered with distinct integers between 1 and n. Two trees are considered different if there is an edge (u, v) which is present in exactly one of the trees.

Since the number of plausible trees can be very large, you have to calculate it modulo 998 244 353.

Input

The only line contains an integer n, the number of vertices in a tree ($1 \le n \le 10^6$).

Output

Print the number of plausible trees with n vertices modulo 998 244 353.

Examples

standard input	standard output
1	0
2	1
3	0
4	12
7788	178152092

Problem I. Binary Supersonic Utahraptors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Alexey and Boris are playing a game called *Binary Supersonic Utahraptors* (BSU).

Initially, Alexey has n utahraptors, and Boris has m utahraptors. Each utahraptor is either yellow or red.

Then, the players take k turns described by integers $s_1, s_2, \ldots, s_k$. The i-th turn is performed as follows. First, Alexey chooses s_i utahraptors that belong to him and gives them to Boris. Then, Boris chooses s_i utahraptors that belong to him (the utahraptors that Alexey has just given to him may also be chosen) and gives them to Alexey.

When the k moves are done, the score of the game is calculated. The score is equal to $|a_y - b_r|$, where a_y is the number of yellow utahraptors Alexey has, and b_r is the number of red utahraptors Boris has. Alexey's goal is to minimize the score, and Boris wants to maximize it.

Write a program that calculates the score of the game if both players use their optimal strategies.

Input

The first line contains three integers n, m, k, the number of utahraptors obtained by Alexey, the number of utahraptors obtained by Boris, and the number of turns in the game ($1 \le n, m, k \le 3 \cdot 10^5$).

The second line contains n integers a_i, denoting Alexey's utahraptors ($0 \le a_i \le 1$). If $a_i = 0$, then the i-th utahraptor is yellow, otherwise the i-th utahraptor is red.

The third line contains m integers b_i, denoting Boris's utahraptors in the same manner as described above ($0 \le b_i \le 1$).

The fourth line contains k integers s_i, describing the numbers of utahraptors that players give to each other on the i-th turn ($1 \le s_i \le \min\{n, m\}$).

Output

Print the score of the game if both players play optimally.

Example

standard input	standard output
2 3 1 0 0 1 1 1 2	1

Problem J. Burnished Security Updates

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Alexander is going to install an important update package called *Burnished Security Updates* (BSU) on his computers. He owns a network which consists of n computers connected by m bidirectional cables.

Eventually, BSU will be installed on every computer in the network. But Alexander doesn't know how the system will behave after the update, so he will first install the update on some non-empty set of computers that satisfies the following conditions:

- no two updated computers are connected directly by a cable;
- each cable must have at least one updated computer as its endpoint;
- the set of the updated computers must be as small as possible.

Formally speaking, if we represent the computer network as a graph, Alexander wants to find an independent set of the graph such that it forms a vertex cover of the same graph. Among all possible sets, he wants to choose one with the least possible size.

Now, you need to help Alexander and find the number of computers on which BSU will be installed. Note that sometimes it can be impossible to find a set satisfying the conditions above at all.

Input

The first line contains two integers n and m, the number of computers and the number of cables ($2 \le n \le 3 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$).

Each of the following m lines contains two integers x_i and y_i, the endpoints of the i-th cable ($1 \le x_i, y_i \le n$, $x_i \ne y_i$).

It is guaranteed that each pair of computers is connected by no more than one cable.

Output

If there is no such set, print a single integer -1.

Otherwise, print the size of a required set of computers.

Examples

standard input	standard output
4 2 1 2 3 4	2
4 4 1 2 2 3 3 4 1 4	2
4 3 1 2 2 3 1 3	-1

Problem L. Business Semiconductor Units

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Business Semiconductor Units (BSU) is a large international corporation that focuses on selling fast and reliable computers to business clients. Recently, they have decided to develop a new processor model which will work even faster and more reliably than its predecessors.

The R&D department of the company is responsible for designing the instruction set and processor architecture. After the deadline, they should demonstrate the working prototype to the head of the company. Unfortunately, the whole department was playing *Minecraft* most of the time instead of doing their job, so the presented prototype supports only three simple instructions.

Let's take a closer look at their masterpiece. The new processor has 16 registers named from `r0` to `r15`, each of them can store an unsigned 16-bit integer. There is also main memory consisting of $2^{16} + 1$ eight-bit cells.

The *program* for this processor is a sequence of instructions. The instructions are executed sequentially, neither jumps nor loops are supported. The processor executes the same sequence of instructions 5000 times. That is, the following procedure is repeated 5000 times: go over the instructions from the start to the end and execute them.

Below you can see the list of available instructions. For clarity, let's call x mod 2^8 the *lower* part of the number x, and $\lfloor \frac{x}{2^8} \rfloor$ the *upper* part of the number x. The number in the i-th main memory cell is denoted mem_i.

- `imm r, b`: load the constant number b ($0 \le b < 2^{16}$) into the register named r;
- `ld x, y`: suppose that the register named y stores the number b. Then, the number $mem_{b+1} \cdot 2^8 + mem_b$ is put into the register x;
- `st x, y`: suppose that the register named x stores the number a, and the register y stores the number b. Then, the lower part of b is put into mem_a, and the upper part of b is put into mem_{a+1}.

As you can see, the instruction set is pretty lean, and the R&D department is unsure whether the processor is capable of doing anything non-trivial or not. To make it run some useful programs, they hired you and gave you an assignment. Now, you need to write a program for the new processor that multiplies n non-negative 16-bit numbers modulo 2^{16}.

Input

This problem has no input data.

Output

Output the required program in the following format.

The first line must contain an integer s, the number of instructions in your program ($1 \le s \le 10^5$).

Each of the following s lines must contain a processor instruction. The format of instructions is described above. Be careful and follow the format strictly. All the register names must be valid (that means, from `r0` to `r15`).

Interaction Protocol

Technically, this problem is an output-only interactive problem. [Sounds weird, isn't it? :)]

In each test, the interactor first reads the instructions you wrote to the output. Next, it reads the integer n and n integers a_i from the test ($1 \le n \le 4000$, $0 \le a_i < 2^{16}$). The number n is placed into the register `r0`. For each $1 \le i \le n$, the lower part of a_i is placed into $mem_{2 \cdot i - 2}$, and the upper part is placed into $mem_{2 \cdot i - 1}$. All other registers and memory cells are zeroed initially.

Then, the interactor executes your program. The instructions are performed sequentially, and the execution of the program is performed exactly 5000 times.

After that, the interactor reads your answer from the register `r0` and compares it with $a_1 \cdot a_2 \cdot \ldots \cdot a_n$ modulo 2^{16}.

Example

standard input	standard output
	4 imm r3, 42 imm r1, 6 st r3, r1 ld r0, r3

Note

The output in the example does not multiply integers, so submitting this program will give you the "`Wrong Answer`" verdict. The program is only provided to illustrate the output format.

Problem M. Brilliant Sequence of Umbrellas

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Anton has n umbrellas, each of them has a different number from 1 to n written on it. He wants to arrange some of the umbrellas in line so that they would form a *brilliant sequence of umbrellas* (BSU). A sequence of k umbrellas with numbers $a_1, a_2, \ldots, a_k$ is considered a BSU if the following rules apply:

- $a_i > a_{i-1}$ for all $2 \le i \le k$;
- $\gcd(a_i, a_{i-1}) > \gcd(a_{i-1}, a_{i-2})$ for all $3 \le i \le k$. Here, $\gcd(x, y)$ denotes the greatest common divisor of integers x and y.

Anton would like to create a long BSU. Making the longest one doesn't bother him, he thinks that a BSU of length at least $\lceil \frac{2}{3}\sqrt{n} \rceil$ is quite enough.

Anton is busy reading fascinating books about lighthouses, so he asks you to find a BSU that would satisfy him.

Input

The only line contains an integer n, the number of umbrellas ($1 \le n \le 10^{12}$).

Output

The first line should contain an integer k, the length of the BSU you have found ($\lceil \frac{2}{3}\sqrt{n} \rceil \le k \le 10^6$).

The second line should contain k integers a_i, the sequence itself ($1 \le a_i \le n$). The sequence should satisfy the rules mentioned above.

Examples

standard input	standard output
10	3 1 2 6
22	4 1 2 6 15

Note

In the first example, $\lceil \frac{2}{3} \cdot \sqrt{10} \rceil = 3$, $\gcd(2, 4) = 2$, $\gcd(4, 8) = 4$.

In the second example, $\lceil \frac{2}{3} \cdot \sqrt{22} \rceil = 4$, $\gcd(1, 6) = 1$, $\gcd(6, 14) = 2$, $\gcd(14, 21) = 7$.

Problem N. Bad Sets Usage

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

In preparation for remodeling of your flat, you're clearing out the junk that has accumulated over the years and find, among other forgotten things, two chess sets you had received as presents. Unfortunately, neither of the sets is complete... In the end you don't have the heart to throw them out and you decide to see if you can assemble one full set from the two partial ones.

Recall that a complete chess set consists of 32 pieces, 16 white and 16 black. There's a king, a queen, two bishops, two knights, two rooks and eight pawns in each color. In the following, the pieces will be named by color and rank, separated by a single space, for example "`white king`", "`black pawn`" etc.

Which pieces from the second set have to be added to the first to get one full set?

Input

The first line of input contains two integers k_1 and k_2 ($1 \le k_1, k_2 < 32$), the numbers of figures left in each set. The follwoing k_1 lines list the contents of the first set in arbitrary order and the following k_2 lines after that the contents of the second set.

Output

Output (in arbitrary order) $32 - k_1$ lines listing the figures that have to be moved from the second set to the first. If a complete set can't be collected, output "`impossible`" as the first and only line.

Examples

standard input	standard output
30 10 black knight white rook white rook black pawn black pawn white pawn black king white queen white bishop black pawn black bishop black rook white knight black pawn white pawn white pawn white pawn white pawn white pawn white pawn white pawn white pawn black pawn white knight white king black pawn black pawn black knight black rook black bishop white bishop black knight white rook white rook black pawn black pawn white pawn black king black queen white bishop black pawn	black queen black pawn
3 1 black knight white rook white rook black pawn	impossible

Problem O. Blots Spilled Unluckly

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Some ink was spilled on a grid paper measuring $M \times N$ cells. Each cell of the paper is now considered either pained or clean. Two painted cells belong to the same blot if there is a path from one of the cells to the other that goes through painted cells only and on each step moves from one cell to another only horizontally or vertically.

Count the number of blots and the area of the largest blot (that is, the number of cells it consists of).

Input

The first input line contains the integers M and N ($1 \le M, N \le 10^5, 1 < M \cdot N \le 10^6$). Each of the following M lines consists of N characters 0 or 1, where 0 denotes a clean and 1 a painted cell. There is at least one painted cell.

Output

The only output line should contain two integers: the number of blots and the area of the largest blot.

Examples

standard input	standard output
6 7 1001001 1111011 1001000 1001111 0100000 0000000	3 13
4 4 0000 0000 0010 0000	1 1

Problem P. Bored Serving Users?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The early 2000's...

Having just boasted about your extensive experience in designing small-scale home networks, you were asked to take on the job for your apartment block. Now is probably not the time to confess that this is really your first attempt.

So, the network will connect N users. Each of them connects to one of the hubs via a dedicated network cable. The hubs may also be connected to one another with similar cables. Any socket of a hub may be used by either an end user or for a connection to another hub. The final network will have to enable any pair of users to communicate with each other via one or more hubs.

Digging through your piles of old hardware, you found M hubs suitable for the purpose. The i^{th} of those hubs ($1 \le i \le M$) has k_i sockets for network cables.

As you have promised that each end user will only have pay for the cable to connect themselves to the nearest hub, you want to design the network using minimal number of your hubs (so you can keep the rest for future projects).

Are you up to the task?

Input

The first input line contains the integers N and M ($2 \le N \le 1000, 1 \le M \le 300$). The second line contains M integers, the values of k_i ($2 \le k_i \le 48$).

Output

If there is no solution, the only line of output should contain the text `Epic fail`.

Otherwise, the first line of output should contain K, the number of hubs used, and the second line K integers, the numbers of the hubs (they are numbered starting from one in the order they are given in the input).

If there are several solutions, output any one of them.

Examples

standard input	standard output
10 2 48 10	1 1
2 5 2 2 2 2 2	1 5
10 1 9	Epic fail
10 2 5 6	Epic fail
20 2 11 11	2 2 1

Problem Q. Base Structure Under...

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A well-known software development company has been commissioned by the Archaeological Society. One of the modules has to help the archaeologists to process data about the ruins of buildings they have found during their excavations of ancient cities. Development of this module has been assigned to Vasya.

Vasya, being a seasoned programmer, at once noticed that the module would need a database to contain the descriptions of the ruins and the estimated construction times of the buildings. It would be all fine, but suddenly the manager got the genial idea that since the database describes Roman ruins, the years of construction should be stored in the Roman number system. Now Vasya is wondering how many symbols he needs to set aside for each year number field in the database. According to the functional specification, the software module must be able to handle years from A to B (inclusive). Help Vasya determine the minimal number of characters sufficient for storing any year number in the range from A to B.

Input

The only input line contains the descriptions of the years A and B, separated by the "-" sign. A description of a year consists of one to four decimal digits (the number of the year), followed by either "AD" (*Anno Domini*, the current era) or "BC" (*Before Christ*, before the current era). In both directions the years are numbered starting from 1. It is known that $753\text{BC} \le A \le B \le 2012\text{AD}$.

Output

The output should consist of a single integer, the minimal number of characters that have to be reserved in the database for the year number.

Examples

standard input	standard output
753BC-747BC	3
1BC-1AD	7
2000AD-2012AD	10

Note

It is well known that the Roman number system uses the following seven letters for digits: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, and M = 1000. Natural numbers are written by repeating these digits. For correct representation of a larger integer in the Roman system, first the thousands, then the hundreds, then the tens and finally the ones are written. In any of the positions, some digits (I, X, C, M) may be repeated, but no more than three times.

If in any position a digit with lower or equal value is to the right of a digit of higher value, the lower value is added to the higher one. Only I, X, C, and M may be added and there may be no more than three equal-valued digits in any position. If, however, a digit with lower value is to the left of a digit with higher value, the lower value is subtracted from the higher one. There are only six subtractions allowed: IV = 4, IX = 9, XL = 40, XC = 90, CD = 400, and CM = 900. Any other combinations are not allowed. So, for example, 99 has to be written as XCIX, not as IC.

It is also important to remember that the Romans did not use the AD/BC concepts. Instead, they counted the years from the mythical founding of Rome (*Anno Urbis Conditae*—753BC).

Some examples of Roman year numbers:

- 753 BC = 1 AUC = `I`
- 1 BC = 753 AUC = `DCCLIII`
- 1 AD = 754 AUC = `DCCLIV`
- 2012 AD = 2765 AUC = `MMDCCLXV`