Problem A. Athletic competition

Input file: `athletic.in`
Output file: `athletic.out`

Elections are coming and the mayor of Bytetown decided to organize an athletic competition to increase his odds. Unfortunately there is no athletic stadium in Bytetown, so all parts of the competition will have to be held in the town's streets. Mayor decided to limit the competition only to running events and asked you for help with planning their tracks.

There are n street junctions and $n-1$ streets in Bytetown. From each junction there is a path leading to every other junction. Every running track must start at some junction, lead along some streets and finish at some other junction. Mayor would like to organize as many events simultaneously as possible, so the number of tracks should be maximized. Due to safety reasons no two tracks may have a common junction or street. Additionally, not every junction can be used as a starting or ending point of an event.

Input

The first line of the input file contains a single integer n ($1 \le n \le 200\,000$), denoting the number of junctions in Bytetown. The junctions are numbered from 1 to n. The following $n-1$ lines contain descriptions of the streets; each of them consists of two integers a and b ($1 \le a, b \le n$, $a \ne b$), separated by a single space — numbers of junctions that are connected by the street.

Line number $n+1$ contains one integer m ($0 \le m \le n$), denoting the number of junctions that can serve as a starting or ending point of an event. The last line of the input file contains m distinct space–separated numbers — the numbers of those junctions, each in the range $[1, n]$.

Output

The first and only line of the input file should contain one integer, denoting the maximal possible number of running events that can be held in Bytetown simultaneously.

Example

athletic.in	athletic.out
5 1 2 3 2 4 1 1 5 4 1 2 5 4	1

Problem B. Cylinders

Input file: `cylinders.in`
Output file: `cylinders.out`

Byteman urgently needs to measure l milliliters of water, so he went to a shop that sells technical glass. Unfortunately, water measuring glass is out of date now and the shop was being wound up at the moment. Most of the glass was sold out and there were only two identical defective cylinders. Their defect was that each of them had only a few of all necessary scales (in milliliters). Byteman had no choice: he needed those cylinders, so he bought them. He is facing a new problem now: how to measure l milliliters of water using his cylinders?

In the beginning both cylinders are empty. Possible actions that can be performed are:

- pouring water from a tap to a cylinder until water reaches one of the cylinder's scales,
- pouring water from a cylinder to the sink until water reaches one of the cylinder's scales,
- pouring water from one cylinder to the other until water reaches one of the first cylinder's scales,
- pouring water from one cylinder to the other until water reaches one of the second cylinder's scales.

Each of these actions takes the same amount of time. Byteman is in a hurry, so he would like to measure the necessary amount of water as quickly as possible. Your task is to either compute the minimal number of actions that Byteman needs to perform in order to obtain exactly l milliliters of water in one of his cylinders or check that the measurement is impossible.

Input

The first line of the input file contains one positive integer n ($3 \le n \le 25$), denoting the number of scales in each of the cylinders. The second line contains an increasing sequence of integers $x_1, \ldots, x_n$ (separated by single spaces) denoting the scales' values. x_1 is always equal to 0 and x_n ($x_n \le 100\,000$) is equal to the capacity of each of the cylinders. The third and last line of the input file contains one integer l ($0 \le l \le x_n$).

Output

If Byteman cannot measure the desired amount of water, your program should output a single word `IMPOSSIBLE`. Otherwise your program should output the minimal number of actions that are needed to perform the measurement.

Example

cylinders.in	cylinders.out
4 0 2 5 15 9	5

Problem C. General Bytor

Input file: `general.in`
Output file: `general.out`

'Qbits are coming!'

General Bytor, the commander-in-chief of Fort Bytemore, woke up, quickly went to the headquarters and looked at the operational plan. Unfortunately the situation didn't look well. In every important strategic position there was one army unit of the Fort; however some units were located at incorrect positions. Even worse, it appeared that there would be huge problems with giving orders. The reason for that was that Qbits' secret agents used quantum teleportation techniques to kidnap all cryptographers from Bytemore! Without them, Bytor won't be able to give all orders that he would like to give, but only just a few of them that he remebered from recent training. Each of them refers to a 'line' leading through a sequence of important strategic positions; if such an order is given, then each army unit on the line moves forward to the next position along the line. Every line is really a cycle, so after the movement is performed there is also a single army unit in every strategic position.

Bytor ordered his spies to find the locations of the Qbits' army units. When the spies come back, there will only remain about half an hour to the beginning of the battle. During this period of time, Bytor will need to give all necessary orders to get all his army units to proper positions. This amount of time will be enough for a moment of thought and performing at most 10 orders. Bytor called for his computer scientists and asked them to write a program that, given the initial and needed army arrangements, checks whether there exists a sequence of orders that is short enough and leads from the initial to the final arrangement.

Input

The first line of the input file contains two integers n and m ($2 \le n \le 75$, $1 \le m \le 10$), separated by a single space. n denotes the number of strategic positions and m is equal to the number of lines.

The second line contains a single n-letter word composed only of small English letters. The i-th letter describes the type of unit located in the i-th strategic point due to the old bytean military code. There can be several units having the same code in the Fort.

The third line also contains a n-letter word. It describes the unit types that should be located in strategic positions $1, 2, \ldots, n$ after the movement. The second word will be different from the first one.

Each of the following m lines contains a description of a 'line' in the form: $c, a_1, a_2, \ldots, a_c$ (the numbers are separated by single spaces). The first number (c) denotes the number of strategic positions in the 'line'. Number a_i ($1 \le a_i \le n$ for $1 \le i \le c$) describes the number of the i-th strategic point in the line. All numbers a_i in a single line will be different. Performing such an order causes the following movement of army units: $a_1 \to a_2$, $a_2 \to a_3$, $\ldots$, $a_{n-1} \to a_n$, $a_n \to a_1$.

Output

If the final arrangement cannot be obtained within 10 orders, your program should output one word `IMPOSSIBLE`. In the opposite case your program should output at most 10 space-separated numbers of 'lines' (between 1 and m), that should be moved in succeeding Bytor's orders. If there are multiple correct solutions, your program should output the one that requires the least number of orders. If there are still multiple solutions, your program should output the lexicographically first (if t is the first position where two solutions of equal lenght differ, then the first of them is lexicographically smaller than the second one iff the t-th order in the first solution has smaller line number than the t-th order in the second solution).

Example

general.in	general.out
12 6 abcdefghijkl cdabghieflkj 2 10 11 4 4 3 2 1 5 9 8 7 6 5 4 1 2 3 4 5 5 6 7 8 9 2 11 12	1 2 2 3 3 6 1

Problem D. Guessing game

Input file: `game.in`
Output file: `game.out`

Byteman is playing a following game with Bitman. Bitman writes down some 1 000 000 000-element sequence of zeros and ones. Byteman's task is to guess the sequence that Bitman created. He can achieve this goal by asking Bitman questions of the form: 'What is the parity of the sum of a subsequence of your sequence, that begins at the b-th element and ends at the e-th element of the sequence?'. After playing the game for some time, Byteman started suspecting that Bitman was cheating. He would like to know at which moment did Bitman first answer his question in an inconsistent way, so he asked you for help.

Write a program which:

- reads a description of Byteman's questions and Bitman's answers,
- computes the greatest number N, for which Bitman's answers for the first N questions are consistent.

Input

The first line of the input file contains one integer n ($0 \leq n \leq 100\,000$), denoting the number of Byteman's questions. Each of the following n lines describes one Byteman's question with corresponding Bitman's answer in the form of three positive integers b, e and s ($1 \leq b \leq e \leq 1\,000\,000\,000$, $s \in \{0, 1\}$), separated by single spaces. b and e are the indices of the first and the last element of the subsequence in Byteman's question. $s = 0$ means that Bitman answered that the sum was even and $s = 1$ — that it was odd.

Output

The first and only line of the output file should contain one integer — the greatest value of N such that there exists a sequence of zeros and ones that is consistent with Bitman's answers to first N Byteman's questions.

Example

game.in	game.out
5 3 3 0 2 5 1 1 4 0 2 5 0 1 5 1	3

An example of a sequence consistent with three first Byteman's questions from the sample input is 01011.

Problem E. Medical examination

Input file: `medical.in`
Output file: `medical.out`

n patients are having a medical examination at a health center today. Each of them received from the center:

- his/her number (a unique integer in the range $1 \dots n$),
- the time when she/he should come to the health center,
- an ordered list of rooms where he/she should come for medical examinations.

The rooms in the health center are numbered with integers $1 \dots m$.

The rules of the center clearly specify the order in which the patients should be examined:

- If the patient's coming time specified by the health center is t, then she/he should come to the first room on his/her list at exactly time t.
- If more than one patient comes for an examination at the same time, then all of them create a queue ordered by their numbers. If there already was a queue in front of the room, then the new queue is appended to the back of the existing one.
- If there is a non-empty queue in front of the room x at time t (for all $t = 0, 1, 2, \dots$), then the first person on the queue enters the room. Examination isn't very thorough, so the person manages to get to the next room on his/her list at time $t + 1$ and the next person from the queue in front of room x may enter that room at time $t + 1$ (if there is any such person).
- If the examination at time t in room x is the last examination on a patient's list, then the patient leaves the health center at time $t + 1$.

Your task is to count at what time will the last patient leave the health center.

Input

The first line of the input file contains two integers n and m ($1 \le n, m \le 1\,000$), separated by a single space and denoting the number of patients that are going to come to the health center and the number of rooms in the center. Each of the following n lines has the form:

$$t_i \ k_i \ g_{i,1} \ g_{i,2} \ \dots \ g_{i,k_i} \text{ (for } 1 \le i \le n).$$

Such line means that patient with number i should come to the center at time t_i ($0 \le t_i \le 1\,000\,000$) and should be examined in k_i ($k_i > 0$) rooms in order: $g_{i,1}, g_{i,2}, \dots, g_{i,k_i}$ ($1 \le g_{i,j} \le m$). All numbers in such line are separated by single spaces. You can assume that $k_1 + k_2 + \dots + k_n$ is not greater than 100 000.

Output

The first and only line of the output file should contain one integer, denoting the time when the last patient will leave the health center.

Example

medical.in	medical.out
5 3 1 3 3 2 1 0 7 2 3 1 1 1 1 2 2 1 1 1 2 3 3 4 3 1 1 1	12

Problem F. NumberEater

Input file: `numbereater.in`
Output file: `numbereater.out`

The NumberEater is a famous monster from Byteland. It eats numbers, but it is very picky: each day its meal must be unique. The monster is given a sequence of integers (a_n). It chooses the starting and ending positions of a meal — i and j $(1 \le i \le j \le n)$ — and prepares a meal that consists of elements $a_i, a_{i+1}, \ldots, a_j$. Two meals $[i_1, j_1]$ and $[i_2, j_2]$ are considered identical by the monster iff the sets of numbers that they contain are the same; in other words:

$$\{a_k : i_1 \le k \le j_1\} = \{a_k : i_2 \le k \le j_2\}$$

Help NumberEater and count the number of different meals that it can eat using only sequence a.

Input

The first line of the input file contains one integer n $(1 \le n \le 500)$, denoting the length of sequence a. The following n lines contain values of elements of sequence a. Each of them is not less than 1 and not greater than 500.

Output

The first and only line of the output file should contain the number of different meals that NumberEater can eat.

Example

numbereater.in	numbereater.out
6 1 2 3 1 2 3	7

Problem G. Nuts

Input file: `nuts.in`
Output file: `nuts.out`

In the very middle of the Hundred Acre Wood there is a huge oak. Squirrel Barbara lives in one of this oak's hollows. Each day she walks along the forest in a quite unusual way. She starts her walk near the oak, facing North, and in each unit of time she performs a move in the following way:

- If there is a nut in her current location then she picks it up, turns 90° right and walks straight for 1 meter.
- If there is no nut where Barbara is standing then she drops a nut in her current location, turns 90° left and also walks straight for 1 meter.

Your task is to count the number of nuts that are lying on the ground after t units of time.

Input

The first line of the input file contains one integer n ($0 \le n \le 7$), denoting the number of nuts that were lying on the ground at time 0. The following n lines describe the nuts' locations: each of them contains two integers x and y ($-2 \le x, y \le 2$), separated by a single space. Such line means that the nut lies x meters to the East and y meters to the *South* from the oak. You can assume that no two nuts lie in the same location. The last line of input contains one integer t ($0 \le t \le 10^9$), denoting the number of units of time that passed since Barbara started her walk.

Output

The first and only line of the input file should contain one non-negative integer, denoting the number of nuts that are lying on the ground after t units of time. We assume that Barbara has an unlimited number of nuts.

Example

nuts.in	nuts.out
2 0 0 -2 1 8	6

Problem H. Ritual

Input file: `ritual.in`
Output file: `ritual.out`

BAB (Bytean Algorithmic Bureau) is a private computer science agency that helps wealthy men solve their confidential problems, which solutions need some usage of computers. Currently BAB was invited by the Grand Master of the Bytoo Worship. On their way to the Bytoo Temple BAB managed to collect some information about the worship.

The Bytoo Worship started when the Prophet received the great silicon slab with two numbers written on it: The Great Sacred Number and The Small Sacred Number. [...] Due to the ritual, each day priests write The Great Sacred Number on the inner side of the wall of the Temple. Priests erase some of its digits during the day using some secret rule and leave the remaining digits on the wall for the night. On the next day The Great Sacred Number is written on the wall again and the ritual is repeated.

The Grand Master of Bytoo gave BAB a preliminary test: they needed to figure out the secret rule of erasing digits. On the first evening BAB noticed that the remaining digits form a palindromic number: the first digit is the same as the last one, the second digit is the same as the last but one and so on. That was however only a part of the secret rule. After a few days BAB used the Euclid's algorithm to discover the second part of the rule: the number that remains after the digits are erased must be divisible by the number of days in the Bytoo week, which is equal to The Small Sacred Number 666. The remaining number may have leading and trailing zeros, but it must have at least one digit (the priests mustn't erase all digits at a time).

The Grand Master congratulated BAB on their discoveries and revealed their real mission. Every day the priests must erase a different configuration of digits. On the first day when the priests won't be able to find any new configuration there will be the end of world. The Grand Master would like to know what day of week will be the last day of the world (priests erased the first configuration of digits on the first day of the Bytoo week).

Input

The first line of the input file contains one integer number G ($1 \le G \le 666^{36}$) — The Great Sacred Number.

Output

The first and only line of the output file should contain one digit from the set $\{1, 2, \ldots, 666\}$, denoting the day of week when the world will end.

Example

ritual.in	ritual.out
6666666	42

There are 35 ways to erase four digits of The Great Sacred Number, leaving number 666 and 7 ways to erase one digit, leaving 666 666. Both 666 and 666 666 are obviously palindromes divisible by 666.

Problem I. Stars

Input file: `stars.in`
Output file: `stars.out`

The Astrologers Association (AA) has recently discovered that the future of Byteland strongly depends on all events (flying meteors, explosions of supernovas) that happen inside the 2-dimensional Woodworm Constellation. This constellation consists of one Alpha star and some Omega stars. Omega stars are vertices of a polygon; this polygon (including its boundary) is called *the interior* of the constellation. Woodworm Constellation has a following property: all segments that connect the Alpha star with points from the constellation's interior lie completely inside the constellation (in its interior).

You are given a list of events that have happened recently in the galaxy. Help AA and determine how many of them have happened inside the Woodworm Constellation.

Input

The first line of the input file contains two integers n and m ($3 \le n \le 100\,000$, $0 \le m \le 100\,000$), separated by a single space and denoting the number of Omega stars and the number of events that have happened in the galaxy recently. Each of the following n lines contains two space-separated integers x_i and y_i ($-32\,000 \le x_i, y_i \le 32\,000$ for $1 \le i \le n$) — coordinates of the i-th Omega star. These coordinates are given in the same order as they lie on the polygon's boundary. The polygon is simple, has positive area and no two stars coincide. We assume that the Alpha star has coordinates $(0, 0)$.

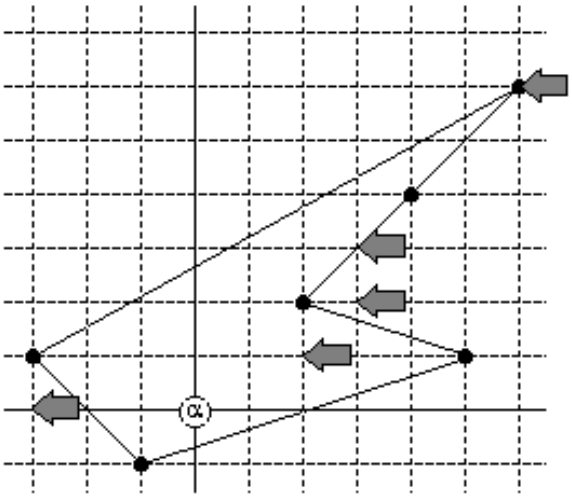
The last m lines of the input file contain descriptions of events that have happened in the galaxy. Each of them is described as a pair of integers x'_j, y'_j ($-32\,000 \le x'_j, y'_j \le 32\,000$ for $1 \le j \le m$), separated by a single space.

Output

The first and only line of the output file should contain the number of events that have happened inside the Woodworm Constellation.

Example

stars.in	stars.out
6 5 2 2 4 4 6 6 -3 1 -1 -1 5 1 2 1 3 2 6 6 3 3 -3 0	3



This is an illustration of the Woodworm Constellation from the sample input. Small black circles show positions of Omega stars and arrows point to events that have happened in the galaxy (3 of them have happened inside the constellation).

Problem J. Sum of a subsequence

Input file: `sum.in`
Output file: `sum.out`

We are given a sequence $a_1, a_2, \ldots, a_{2n}$. Your task is to find a subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$ of this sequence such that its sum $a_{i_1} + a_{i_2} + \cdots + a_{i_n}$ is divisible by n, or check that such subsequence doesn't exist.

Input

The first line of the input file contains one integer n ($1 \le n \le 100\,000$). The second line contains $2n$ integers a_i ($1 \le a_i \le 1\,000\,000\,000$), separated by single spaces. You can assume that $n \in \{10, 100, 1\,000, 10\,000, 100\,000\}$.

Output

If there is no subsequence that satisfies the described conditions then the first and only line of the output file should contain one word `IMPOSSIBLE`. Otherwise the first and only line of the output file should contain n integers $i_1 < i_2 < \cdots < i_n$, such that $n|(a_{i_1} + a_{i_2} + \cdots + a_{i_n})$. In case of multiple correct answers your program should output any one.

Example

sum.in	sum.out
10 17 10 13 15 10 4 14 4 9 5 3 2 1 5 19 12 19 9 17 11	1 3 5 6 12 13 14 16 18 19

The real sample input fits in exactly two lines — the example above is formatted in 3 lines only for technical reasons.

Problem K. Table game

Input file: `table.in`
Output file: `table.out`

n children sit at a round table. Child number 1 (that happens to be Jessica) has n candies and all remaining children (numbered as they sit along the table: $2, \ldots, n$) have no candies. Children are playing a game, which purpose is to:

- distribute the candies evenly among the children and
- have fun!

The game consists of *moves*. During each move, some child, that has at least two candies at the moment, gives one candy to each of its two neighbours at the table. The game ends when each child has exactly one candy. Help the children and output the shortest sequence of moves that leads to an even distribution of candies.

Input

The first line of the input file contains one integer number n ($3 \leq n \leq 300$), that represents both the number of children sitting at the table and the number of Jessica's candies.

Output

If there is no valid sequence of moves, your program should output exactly one word `IMPOSSIBLE`. In the opposite case the output file should contain a sequence of space–separated numbers $a_1, a_2, \ldots$ Value $a_i = k$ means that in the i-th move child number k ($1 \leq k \leq n$) gives one candy to each of its neighbours. After the sequence of operations is performed, each child should have exactly one candy.

If there is more than one valid shortest sequence of moves, your program should output any one.

Example

table.in	table.out
5	1 1 2 5 1

The configurations of candies at the table during this game (numbers represent quantities of candies that children $1, 2, 3, 4, 5$ have at the moment) are:

- 5 0 0 0 0
- 3 1 0 0 1
- 1 2 0 0 2
- 2 0 1 0 2
- 3 0 1 1 0
- 1 1 1 1 1