Problem A. Linear Device

Input file:	`linear.in`
Output file:	`linear.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Research Institute of Graph Handling Theory (RIGHT) needs to build a special device to support their new experiment. The scheme of this device contains three types of elements: *generators*, *resistors* and *transistors*. Each **generator** is connected with exactly **one** other element, **resistor** is connected with **two** other elements, and **transistor** is connected with **three** of them. There is also a special condition, that **no transistor is connected to another transistor**.

All elements must be positioned on a metal string in such way that any two subsequent elements must be connected. Two non-subsequent elements are always allowed to be connected by an extra wire. To position all elements on a string in such a way, specialists of RIGHT invited Vasya from RIGS (Research Institute of Given Strings). But this is not a string Vasya used to deal with... so he asked you for help.

Input

The first line contains the number of elements in the device N ($2 \leqslant N \leqslant 100\,000$). The next N lines contain descriptions of elements and their connections. Each of them starts with a single letter `G`, `R` or `T` which stands for Generator, Resistor or Transistor respectively. One, two or three numbers follow, separated by spaces, indicating the numbers of elements connected with the corresponding element. Elements are enumerated from 1 to N in order of their description in input. Element cannot be connected to itself, and two elements are not allowed to be connected more than once.

Output

The only line of the output must contain N integer numbers — a permutation of elements that allows to position them all on a metal string such that any two consecutive elements are connected. If there is more than one solution, any solution is accepted. If there is no solution, output a single word `IMPOSSIBLE`.

Examples

linear.in	linear.out
5 G 2 R 1 3 T 2 4 5 R 3 5 R 3 4	5 4 3 2 1
7 T 2 3 4 G 1 G 1 R 1 7 G 7 G 7 T 4 5 6	IMPOSSIBLE

Problem B. The Queen and The King

Input file:	`queen.in`
Output file:	`queen.out`
Time limit:	0.3 seconds
Memory limit:	64 megabytes

The chess Queen and the chess King want to capture each other. Today they have met on the chessboard $2 \times n$ with some walls between its fields. Your task is to determine the winner again!

At start the King is positioned on the field $(1, 1)$ and the Queen stands on $(2, n)$. The King moves first.

Both sides act in an optimal way.

The side is considered a loser when it has no moves available. The moves after which a piece can be captured by other piece are not allowed.

The pieces cannot move through the walls. In the case of diagonal move it is not allowed when any one of the four possible walls touching the corner is present.

Input

The first line of the input file contains two integer numbers: n and m ($2 \leqslant n \leqslant 50$, $m \leqslant 300$). m lines follow, containing each four integer numbers x_1, y_1, x_2, y_2 — coordinates of the adjacent fields separated by the wall ($1 \leqslant x_i \leqslant 2$, $1 \leqslant y_i \leqslant n$). The walls are only horizontal and vertical. A single wall may be described in input file more than once.

Output

If the King has winning moves, the first line contains `king wins`. If the Queen will be a winner, the first line contains the message `queen wins`. The second line must be the number of moves the game will last if both sides play optimally.

In the case of a draw, write just the single line `draw`.

Example

queen.in	queen.out
2 1 1 2 2 2	queen wins 2
2 2 1 2 2 2 2 2 2 1	king wins 1
2 2 1 2 2 2 1 1 2 1	draw

Problem C. Result of the Game

Input file:	`result.in`
Output file:	`result.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Here we will consider a game very similar to the New Game invented by Sasha in previous task. There will be N states and some arrows between them. But there will be two differences: firstly, there will be no scores written on arrows, and secondly, there will be no cycles at all.

Imagine that two players are playing this game moving one piece in turn. If somebody can't move, (s)he loses. So the goal of the game is to set your piece in a position from which your opponent won't be able to move.

There are some very useful theoretical facts about the games of this kind. Most of them involve the so-called *Sprague–Grundy numbers*. These numbers are defined as follows. For position X without outgoing moves we define $SG(X) = 0$. For any other position Y we put $SG(Y) = \min\{t \in \mathbb{Z}, t \geqslant 0 : t \neq SG(Z)\ \forall Z : Y \to Z\}$, where $Y \to Z$ means that there is a move from Y to Z. So the Sprague–Grundy number of X is defined to be the minimal non-negative integer not occuring in the set $\{SG(Y)\}$ where Y runs through all positions obtained from X in one move.

The Sprague–Grundy numbers are very useful because they can be

used to analyze *direct sums* of games. Imagine that we have two games $\mathcal{G}$ and $\mathcal{H}$ with a piece in each game. A player can choose any game to make his/her next move. The resulting game can be described as one game $\mathcal{G} \oplus \mathcal{H}$ with the set of states equal to $S_\mathcal{G} \times S_\mathcal{H}$ where $S_\mathcal{G}$ are states of $\mathcal{G}$ and $S_\mathcal{H}$ are states of $\mathcal{H}$. The surprising fact is that there is a smart way to calculate Sprague–Grundy numbers for this game. Namely, $SG(X_1, X_2) = SG(X_1) \oplus SG(X_2)$, where $X_1 \in S_\mathcal{G}$, $X_2 \in S_\mathcal{H}$, and $\oplus$ means bitwise exclusive OR (XOR). The same is true for direct sums of more than two games: if we have k games and can select any one of them to play before each move, then $SG(X_1, X_2, \ldots, X_k) = SG(X_1) \oplus SG(X_2) \oplus \cdots \oplus SG(X_k)$.

Another surprising fact making these numbers useful is that $SG(X)$ is non-zero if and only if the player starting from position X has a winning strategy. Moreover, the winning move can be determined by considering $SG(Y)$ for all Y such that $X \to Y$ and choosing any Y with $SG(Y) = 0$ as the next move.

Your task is to calculate Sprague–Grundy numbers for a given game.

Input

The first line of the input contains the number of states in the game N ($1 \leqslant N \leqslant 5\,000$) and the number of possible moves M ($1 \leqslant M \leqslant 100\,000$). The next M lines describe one move each, in form x_i y_i which means that there is a move from state x_i to state y_i. Here $1 \leqslant x_i, y_i \leqslant N$. All moves are different.

Output

The j-th line must contain value of $SG(j)$.

Example

result.in	result.out
8 8 1 2 1 8 1 4 8 6 8 7 6 3 7 3 2 3	2 1 0 0 0 1 1 0

Problem D. Sum of Games

Input file:	smith.in
Output file:	smith.out
Time limit:	1 seconds
Memory limit:	256 mebibytes

The game on arbitrary directed graph is played as follows: at the beginning there is a pawn in one of the graph vertices (this vertex is called a starting vertex). Two players move in turn, one move is to select an edge from the current vertex and move the pawn to the other end of it. The player who can't move loses.

Sometimes people consider some combinations of such games. For example, the nim-sum is the game consisiting of two games, each is one graph with a pawn on it, and the player can select exactly one of the pawns to move.

Your task is to determine the winner assuming both players play optimally.

Input

First line contains two integers N_1 and M_1 — the number of vertices and edges in the first graph ($1 \leqslant N_1$, $M_1 \leqslant 10\,000$). M_1 lines follow, each containing two numbers x and y ($1 \leqslant x$, $y \leqslant N_1$) — the edge from x to y. Loops and multiple edges are allowed.

The next $M_2 + 1$ contain the second graph in the same format.

The last lines describe the pairs of starting vertices for which you should determine the winner. First of them contains a single integer T ($1 \leq T \leq 100\,000$) — the number of queries. T lines follow, each containing two vertices v_1 and v_2 ($1 \leqslant v_1 \leqslant N_1$, $1 \leqslant v_2 \leqslant N_2$).

Output

For each of T queries output a single line"`first`", if the first player wins, "`second`", if the second one wins, "`draw`", if the game will end in a draw.

Example

smith.in	smith.out
3 2 1 2 2 3 2 1 1 2 2 1 1 3 2	first second

Problem E. Square Root

Input file:	sqroot.in
Output file:	sqroot.out
Time limit:	2 seconds
Memory limit:	64 mebibytes

You are give two integers a and n, $0 \leqslant a < n \leqslant 10^9$. Your task is to find square root of a modulo n, i.e. such x ($0 \leqslant x < n$) that x^2 has the remainder a modulo n.

Input

The input consists of one ore more cases, the number of them $T \leqslant 300$ is given on the first line. On the next T lines there are two integers on each — a and n.

Output

For each of the cases output on a separate line any square root of a modulo n, or `IMPOSSIBLE`, if no roots exist. Square root must be between 0 and $n - 1$, inclusive.

Example

sqroot.in	sqroot.out
2 1 3 1 4	1 1

Problem F. Post Marks

Input file:	stamps.in
Output file:	stamps.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Your hobby is collecting postmarks. There is a total of n distinct postmarks. Their prices are given to you. The price of a postmark is the amount of money you would get for selling it, and the amount of money it would cost you to buy it. Also you are given the values of the postmarks. Only one of each postmark exists.

You currently have some of the postmarks. Your goal is to have a postmark collection with a total value of at least k. You can sell postmarks to get money to buy different postmarks. Find the minimum additional amount of money you need to achieve your goal.

Input

The first line contains two integers n and k ($1 \leqslant n \leqslant 32$, $0 \leqslant k \leqslant 10^9$). The second line contains costs of the postmarks p_1, p_2, ..., p_n. The third line contains n integers h_1, h_2, ..., h_n; h_i is 1, if you already have i-th postmarks, 0 otherwise. The fourth line contains n integers v_1, v_2, ..., v_n — the values of the postmarks. All costs and values are non-negative and do not exceed 30 000 000.

Output

In the first line output the minimal number of money required to get the postmark collection with value at least k. If you can do it without spending any money or even earning some money, output 0. If it is impossible regardless of amount of money, output -1.

Examples

stamps.in
2 13 2 15 0 0 2 21
stamps.out
15

stamps.in
5 67 9 18 7 6 18 1 0 0 0 1 12 27 10 10 25
stamps.out
22

stamps.in
4 10 14 14 12 6 0 1 1 1 19 23 20 7
stamps.out
0

stamps.in
10 811 43 33 14 31 42 37 17 42 40 20 0 0 0 0 0 0 1 0 0 0 116 71 38 77 87 106 48 107 91 41
stamps.out
-1

Problem G. The Word Game

Input file:	`wordgame.in`
Output file:	`wordgame.out`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Two players play the following game with words, which slightly resembles Mastermind:

First player secretly picks a word W and tells its length to second. The second player tries to guess the word, telling to first the words G_i of the same length, and the first player responds by telling whether the number of correct letters in correct positions is even or odd. This is repeated until the second player can figure out what the first player's picked word is.

You are given a list of guesses G_i and your task is to determine possible words W.

Input

The first line contains N and M — number of guesses with even and odd number of correct positions, reqspectively. It is guaranteed that $1 \leq N + M \leq 64$.

The second line contains N words of the same length, consisting of the same number of uppercase English letters — the guesses with even number of correct position, separated by spaces.

The third line contains M words of the same length, consisting of the same number of uppercase English letters — the guesses with odd number of correct position, separated by spaces.

The length of all words are the same, they are always between 1 and 9 characters, inclusive. It is guaranteed K, which is the total number of possible W is not greater than 1000 for each case.

Output

Output the number K in the first line. On the second line output K words separated by spaces — the possible values for W, in lexicographical order.

Examples

wordgame.in	wordgame.out
3 3 DAY MAY BUY SAY DUE TEN	1 SEE