Problem A. Alignment of Code

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

You are working in a team that writes Incredibly Customizable Programming Codewriter (ICPC) which is basically a text editor with bells and whistles. You are working on a module that takes a piece of code containing some definitions or other tabular information and aligns each column on a fixed vertical position, while keeping the resulting code as short as possible, making sure that only whitespaces that are absolutely required stay in the code. So, that the first words on each line are printed at position $p_1 = 1$; the second words on each line are printed at the minimal possible position p_2, such that all first words end at or before position $p_2 - 2$; the third words on each line are printed at the minimal possible position p_3, such that all second words end at or before position $p_3 - 2$, etc.

For the purpose of this problem, the code consists of multiple lines. Each line consists of one or more words separated by spaces. Each word can contain uppercase and lowercase Latin letters, all ASCII punctuation marks, separators, and other non-whitespace ASCII characters (ASCII codes 33 to 126 inclusive). Whitespace consists of space characters (ASCII code 32).

Input

The input file contains one or more lines of the code up to the end of file. All lines (including the last one) are terminated by a standard end-of-line sequence in the file. Each line contains at least one word, each word is 1 to 80 characters long (inclusive). Words are separated by one or more spaces. Lines of the code can have both leading and trailing spaces. Each line in the input file is at most 180 characters long. There are at most 1000 lines in the input file.

Output

Write to the output file the reformatted, aligned code that consists of the same number of lines, with the same words in the same order, without trailing and leading spaces, separated by one or more spaces such that i-th word on each line starts at the same position p_i.

Sample input and output

The '␣' character in the example below denotes a space character in the actual files (ASCII code 32).

alignment.in	alignment.out
␣␣start:␣␣integer;␣␣␣␣//␣begins␣here stop:␣integer;␣//␣␣ends␣here␣␣ ␣s:␣␣string;␣␣␣ c:␣␣␣char;␣//␣temp␣	start:␣integer;␣//␣begins␣here stop:␣␣integer;␣//␣ends␣␣␣here s:␣␣␣␣␣string; c:␣␣␣␣␣char;␣␣␣␣//␣temp

Problem B. Binary Operation

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

Consider a binary operation $\odot$ defined on digits 0 to 9, $\odot : \{0, 1, \ldots, 9\} \times \{0, 1, \ldots, 9\} \to \{0, 1, \ldots, 9\}$, such that $0 \odot 0 = 0$.

A binary operation $\otimes$ is a generalization of $\odot$ to the set of non-negative integers, $\otimes : \mathbb{Z}_{0+} \times \mathbb{Z}_{0+} \to \mathbb{Z}_{0+}$. The result of $a \otimes b$ is defined in the following way: if one of the numbers a and b has fewer digits than the other in decimal notation, then append leading zeroes to it, so that the numbers are of the same length; then apply the operation $\odot$ digit-wise to the corresponding digits of a and b.

$$\begin{array}{r} 5566 \\ \otimes\ 239 \\ \hline ???? \end{array} \longrightarrow \begin{array}{r} 5566 \\ \otimes\ 0239 \\ \hline ???? \end{array} \longrightarrow \begin{array}{cccc} 5 & 5 & 6 & 6 \\ \odot\ 0 & \odot\ 2 & \odot\ 3 & \odot\ 9 \\ \hline 0 & 0 & 8 & 4 \end{array} \longrightarrow \begin{array}{r} 5566 \\ \otimes\ 0239 \\ \hline 0084 \end{array} \longrightarrow \begin{array}{r} 5566 \\ \otimes\ 239 \\ \hline 84 \end{array}$$

Example. If $a \odot b = ab \bmod 10$, then $5566 \otimes 239 = 84$.

Let us define $\otimes$ to be left-associative, that is, $a \otimes b \otimes c$ is to be interpreted as $(a \otimes b) \otimes c$.

Given a binary operation $\odot$ and two non-negative integers a and b, calculate the value of $a \otimes (a+1) \otimes (a+2) \otimes \ldots \otimes (b-1) \otimes b$.

Input

The first ten lines of the input file contain the description of the binary operation $\odot$. The i-th line of the input file contains a space-separated list of ten digits — the j-th digit in this list is equal to $(i-1) \odot (j-1)$.

The first digit in the first line is always 0.

The eleventh line of the input file contains two non-negative integers a and b ($0 \le a \le b \le 10^{18}$).

Output

Output a single number – the value of $a \otimes (a+1) \otimes (a+2) \otimes \ldots \otimes (b-1) \otimes b$ without extra leading zeroes.

Sample input and output

binary.in	binary.out
0 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 0 2 3 4 5 6 7 8 9 0 1 3 4 5 6 7 8 9 0 1 2 4 5 6 7 8 9 0 1 2 3 5 6 7 8 9 0 1 2 3 4 6 7 8 9 0 1 2 3 4 5 7 8 9 0 1 2 3 4 5 6 8 9 0 1 2 3 4 5 6 7 9 0 1 2 3 4 5 6 7 8 0 10	15

Problem C. Cactus Revolution

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

Advanced Cave Megapolis (ACM) is a city that survives in the underground caves after the global nuclear war. The caves are connected by passages and the whole city map can be represented by a graph with caves being vertices and passages between them being nodes.

There is a revolution in the cave city. The whole population of the city is evenly split into k parties that cannot agree on the common laws that they should adopt. They had decided to split their city into k districts and have each district's citizens impose the laws of their liking upon themselves.

You are given a city map in the form of the graph and your task is to write a program that partitions this graph into k equally sized districts. Each district must form a connected subgraph that is represented by the subset of the graph's vertices.

Fortunately, the number of vertices in the graph is divisible by k and the graph representing the city happens to be a *cactus* — a connected undirected graph in which every edge belongs to at most one simple cycle. Intuitively, cactus is a generalization of a tree where some cycles are allowed.

The example of a city map with 15 caves and its partitioning into 3 districts is shown on the picture below.

Input

The first line of the input file contains three integer numbers n, m, and k ($1 \le n \le 50\,000$, $0 \le m \le 10\,000$, $1 \le k \le n$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths, k is the number of districts that the city must be partitioned into, n is divisible by k.

Each of the following m lines contains a path in the graph. A path starts with an integer number s_i ($2 \le s_i \le 1000$) followed by s_i integers from 1 to n. These s_i integers represent vertices of a path. Adjacent vertices in a path are distinct. Path can go through the same vertex multiple times, but every edge is traversed exactly once in the whole input file. There are no multiedges in the graph (there is at most one edge between any two vertices).

The graph in the input file is a cactus.

Output

If it is possible to partition the vertices into k districts, write to the output file k lines with n/k integer numbers on each line. Each line represents a district as a list of vertices' numbers that constitute it. Vertex numbers must be listed in the ascending order in the description of each district.

If the answer does not exist, write the single number -1.

Sample input and output

cactus.in	cactus.out
15 3 3 9 1 2 3 4 5 6 7 8 3 7 2 9 10 11 12 13 10 5 2 14 9 15 10	4 5 6 7 8 10 11 12 13 15 1 2 3 9 14
4 2 2 3 1 2 3 2 2 4	-1

Problem D. Dome of Circus

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

A travelling circus faces a tough challenge in designing the dome for its performances. The circus has a number of shows that happen above the stage in the air under the dome. Various rigs, supports, and anchors must be installed over the stage, but under the dome. The dome itself must rise above the center of the stage and has a conical shape. The space under the dome must be air-conditioned, so the goal is to design the dome that contains minimal volume.

You are given a set of n points in the space; (x_i, y_i, z_i) for $1 \le i \le n$ are the coordinates of the points in the air above the stage that must be covered by the dome. The ground is denoted by the plane $z = 0$, with positive z coordinates going up. The center of the stage is on the ground at the point $(0, 0, 0)$.

The tip of the dome must be located at some point with coordinates $(0, 0, h)$ with $h > 0$. The dome must have a conical shape that touches the ground at the circle with the center in the point $(0, 0, 0)$ and with the radius of r. The dome must contain or touch all the n given points. The dome must have the minimal volume, given the above constraints.

Input

The first line of the input file contains a single integer number n ($1 \le n \le 10\,000$) — the number of points under the dome. The following n lines describe points with three floating point numbers x_i, y_i, and z_i per line — the coordinates of i-th point. All coordinates do not exceed 1000 by their absolute value and have at most 2 digits after decimal point. All z_i are positive. There is at least one point with non-zero x_i or y_i.

Output

Write to the output file a single line with two floating point numbers h and r — the height and the base radius of the dome. The numbers must be precise up to 3 digits after decimal point.

Sample input and output

dome.in	dome.out
1 1.00 0.00 1.00	3.000 1.500
2 1.00 0.00 1.00 0.00 1.50 0.50	2.000 2.000
3 1.00 0.00 1.00 0.00 1.50 0.50 -0.50 -0.50 1.00	2.000 2.000

Problem E. Evacuation Plan

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

Flatland government is building a new highway that will be used to transport weapons from its main weapon plant to the frontline in order to support the undergoing military operation against its neighbor country Edgeland. Highway is a straight line and there are n construction teams working at some points on it.

During last days the threat of a nuclear attack from Edgeland has significantly increased. Therefore the construction office has decided to develop an evacuation plan for the construction teams in case of a nuclear attack. There are m shelters located near the constructed highway. This evacuation plan must assign each team to a shelter that it should use in case of an attack.

Each shelter entrance must be securely locked from the inside to prevent any damage to the shelter itself. So, for each shelter there must be some team that goes to this shelter in case of an attack. The office must also supply fuel to each team, so that it can drive to its assigned shelter in case of an attack. The amount of fuel that is needed is proportional to the distance from the team's location to the assigned shelter. To minimize evacuation costs, the office would like to create a plan that minimizes the total fuel needed.

Your task is to help them develop such a plan.

Input

The first line of the input file contains n — the number of construction teams ($1 \le n \le 4000$). The second line contains n integer numbers — the locations of the teams. Each team's location is a positive integer not exceeding 10^9, all team locations are different.

The third line of the input file contains m — the number of shelters ($1 \le m \le n$). The fourth line contains m integer numbers — the locations of the shelters. Each shelter's location is a positive integer not exceeding 10^9, all shelter locations are different.

The amount of fuel that needs to be supplied to a team at location x that goes to a shelter at location y is equal to $|x - y|$.

Output

The first line of the output file must contain z — the total amount of fuel needed. The second line must contain n integer numbers: for each team output the number of the shelter that it should be assigned to. Shelters are numbered from 1 to m in the order they are listed in the input file.

Sample input and output

evacuation.in	evacuation.out
3 1 2 3 2 2 10	8 1 1 2

Problem F. Factorial Simplification

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

Peter is working on a combinatorial problem. He has carried out quite lengthy derivations and got a resulting formula that is a ratio of two products of factorials like this:

$$\frac{p_1!p_2!\ldots p_n!}{q_1!q_2!\ldots q_m!}$$

This does not surprise Peter, since factorials appear quite often in various combinatorial formulae, because $n!$ represents the number of transpositions of n elements — one of the basic combinatorial objects.

However, Peter might have made a mistake in his derivations. He knows that the result should be an integer number and he needs to check this first. For an integer result Peter wants to simplify this formula to get a better feeling of its actual combinatorial significance. He wants to represent the same number as a product of factorials like this.

$$r_1!^{s_1} r_2!^{s_2} \ldots r_k!^{s_k} t$$

where all r_i are distinct integer numbers greater than one in the descending order ($r_i > r_{i+1} > 1$), s_i and t are positive integers. Among all the possible representations in this form, Peter is interested in one where r_1 is the largest possible number, among those in the one where s_1 is the largest possible number; among those in the one where r_2 is the largest possible number; among those in the one where s_2 is the largest possible number; etc, until the remaining t cannot be further represented in this form. Peter does not care about the actual value of t. He wants to know what is the factorial-product part of his result.

Input

The first line of the input file contains two integer numbers n and m ($1 \le n, m \le 1000$). The second line of the input file contains n integer numbers p_i ($1 \le p_i \le 10\,000$) separated by spaces. The third line of the input file contains m integer numbers q_i ($1 \le q_i \le 10\,000$) separated by spaces.

Output

On the first line of the output write a single integer number k. Write $k = -1$ if the ratio of the given factorial products is not an integer. Write $k = 0$ if the ratio is an integer but it cannot be represented in the desired form. Write $k > 0$ followed by k lines if the ratio can be represented by a factorial product as described in the problem statement. On each of the following k lines write two integers r_i and s_i (for $i = 1 \ldots k$) separated by a space.

Sample input and output

factorial.in	factorial.out
1 2 6 4 4	-1
1 2 6 3 4	0
4 2 9 2 2 2 3 4	2 7 1 2 2

Problem G. Game of 10

Input file: `standard input`
Output file: `standard output`

The Game of 10 is played by two players on a 4×4 field. Initially all 16 cells of the field are empty. Players make alternating moves. On each move a player writes a number from 1 to 4 into an empty cell. The first player that makes any row or column filled with four numbers with a sum of 10 wins. If all cells are filled but no row or column has a sum of 10, then a draw is declared.

You have to write a program that plays for the second player and always wins.

The table below shows the field after the sample game that is shown in the "Sample input and output" section. Subscripts denote the number of the move in the game starting from the first one, with 14-th being the last and winning move by the second player. The last move in the game had filled the second column with a sum of 10.

1_3	4_{13}	2_8	2_{10}
1_7	2_1		2_5
3_{11}	1_{14}	3_2	4_9
3_{12}	3_6		4_4

Interaction protocol

The interaction starts with your program reading the first player's move from the standard input. Then your program must write its move to the standard output, wait for the first player's move in the standard input and so on.

Your program must exit after writing the last, winning move to the standard output. Your program must write end-of-line sequence and flush the standard output after each move, including the last, winning move.

Input

The standard input consists of the first player's moves. Each move is represented by a single line that contains three integer numbers r, c, and k ($1 \le r, c, k \le 4$) separated by spaces, where r and c are row and column numbers, and k is the number that the first player writes into the cell (r, c).

Output

The standard output consists of the second player's moves. Each move is represented by a single line of the same format as in the standard input. The last, winning move, shall have an extra word "`WIN`" (without quotes) on the same line after the three numbers that denote the move, separated from them by a space.

Sample input and output

standard input	standard output
2 2 2 1 1 1 2 4 2 2 1 1 3 4 4 3 1 3 1 2 4	3 3 3 4 4 4 4 2 3 1 3 2 1 4 2 4 1 3 3 2 1 WIN

Problem H. Hands of Poker

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

The standard 52-card deck consists of 52 cards divided into 4 suits: clubs, diamonds, hearts and spades. For each suit there are 13 ranks: 2, 3, 4, 5, 6, 7, 8, 9, 10, jack, queen, king and ace, listed from the lowest to the highest.

A card is denoted by its rank ('`2`'...'`9`' for 2...9, '`T`' for 10, '`J`' for jack, '`Q`' for queen, '`K`' for king, and '`A`' for ace) followed by its suit ('`C`' for clubs, '`D`' for diamonds, '`H`' for hearts, and '`S`' for spades). Cards are partially ordered by their ranks. The suit does not play a role in the cards ordering.

A Poker *hand* is a set of five distinct cards. Each hand is said to have a certain *ranking*. A hand with a higher ranking *beats* a hand with a lower one. Two hands of the same ranking are compared using a tie-breaking rule specific for their ranking — either one of them beats the other or they are tied.

The list of poker rankings is given below, from the worst ranking to the best ranking. If a hand satisfies several rankings, only the best one is considered.

- High Card — Does not fit into any ranking below. When comparing with another High Card hand, the ranks of the highest cards in the two hands are first compared. If there is a tie, the second highest cards in each hand are compared, and so on. (*Example*: QS, JH, 9C, 7H, 3D)
- One Pair — Two cards of the same rank. Pair with higher rank beats the lower pair. In case of a tie, the remaining three cards are used as tie-breakers, compared in the descending order of their ranks (as in High Card). (*Example*: 6D, 6H, QD, 9H, 4S)
- Two Pairs — Two pairs of cards of the same rank. When comparing with another Two Pairs hand, the higher pair is first compared, then the lower pair, and finally the rank of the fifth remaining card. (*Example*: JH, JS, TS, TD, 8S)
- Three of a Kind — Three cards of the same rank. Three-of-a-kind with the higher rank beats the lower one. In case of a tie, the remaining two cards are used as tie-breakers, compared in the descending order. (*Example*: 5S, 5H, 5D, JH, 6D)
- Straight — Five cards in consecutive rank. An ace can either be accounted above a king or below a two, but not both, so wrapping is not allowed. Two straights are compared using the rank of the highest card (in the case of A, 2, 3, 4, 5, the highest card is considered to be 5). (*Example*: QH, JC, TH, 9D, 8D)
- Flush — Five cards of the same suit. When comparing two Flushes, the rank of the highest card is first considered, then the second highest and so on (as in High Card). (*Example*: AS, JS, 8S, 6S, 5S)
- Full House — Three cards of the same rank, and two cards of same rank. When comparing with another Full House, the rank of the three cards is first compared, then the rank of the two cards. (*Example*: 7S, 7H, 7C, JC, JH)
- Four of a Kind — Four cards of the same rank. Two four-of-a-kinds are first compared by the ranks of the four cards. In case of a tie, the rank of the fifth card is used as a tie-breaker. (*Example*: 4C, 4D, 4H, 4S, TD)
- Straight Flush — A hand that is both a Straight and a Flush. Same tie-breaker as for a Straight. (*Example*: TH, 9H, 8H, 7H, 6H)

Consider the set $\mathcal{H}$ of all Poker hands. Let us introduce an evaluation function $v : \mathcal{H} \to \{1, \ldots, 7462\}$, such that for any two Poker hands a and b, a beats b if and only if $v(a) > v(b)$. There exists exactly one such evaluation function v.

Given a Poker hand a, find the value of $v(a)$.

Input

The input file contains space-separated list of five distinct card descriptions. Each card is described with two characters denoting its rank and suit, respectively. The ranks are denoted by '`2`'...'`9`', '`T`', '`J`', '`Q`', '`K`', and '`A`' (listed here in the ascending order). The suits are denoted by '`C`', '`D`', '`H`', and '`S`'.

Output

Output the value of the evaluation function $v(a)$ for the given hand a.

Sample input and output

hands.in	hands.out
3S 7S 2C 4S 5H	1
JH AH TH KH QH	7462

Problem I. Ideal Path

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

New labyrinth attraction is open in New Lostland amusement park. The labyrinth consists of n rooms connected by m passages. Each passage is colored into some color c_i. Visitors of the labyrinth are dropped from the helicopter to the room number 1 and their goal is to get to the labyrinth exit located in the room number n.

Labyrinth owners are planning to run a contest tomorrow. Several runners will be dropped to the room number 1. They will run to the room number n writing down colors of passages as they run through them. The contestant with the shortest sequence of colors is the winner of the contest. If there are several contestants with the same sequence length, the one with the *ideal path* is the winner. The path is the ideal path if its color sequence is the lexicographically smallest among shortest paths.

Andrew is preparing for the contest. He took a helicopter tour above New Lostland and made a picture of the labyrinth. Your task is to help him find the ideal path from the room number 1 to the room number n that would allow him to win the contest.

Note

A sequence $(a_1, a_2, \ldots, a_k)$ is lexicographically smaller than a sequence $(b_1, b_2, \ldots, b_k)$ if there exists i such that $a_i < b_i$, and $a_j = b_j$ for all $j < i$.

Input

The first line of the input file contains integers n and m — the number of rooms and passages, respectively ($2 \le n \le 100\,000$, $1 \le m \le 200\,000$). The following m lines describe passages, each passage is described with three integer numbers: a_i, b_i, and c_i — the numbers of rooms it connects and its color ($1 \le a_i, b_i \le n$, $1 \le c_i \le 10^9$). Each passage can be passed in either direction. Two rooms can be connected with more than one passage, there can be a passage from a room to itself. It is guaranteed that it is possible to reach the room number n from the room number 1.

Output

The first line of the output file must contain k — the length of the shortest path from the room number 1 to the room number n. The second line must contain k numbers — the colors of passages in the order they must be passed in the ideal path.

Sample input and output

ideal.in	ideal.out
4 6 1 2 1 1 3 2 3 4 3 2 3 1 2 4 4 3 1 1	2 1 3

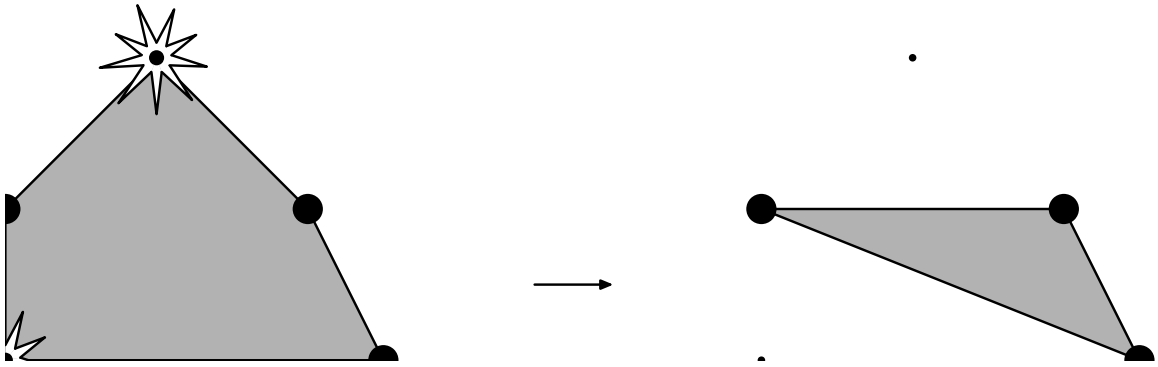
Problem J. Jungle Outpost

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

There is a military base lost deep in the jungle. It is surrounded by n watchtowers with ultrasonic generators. In this problem watchtowers are represented by points on a plane.

Watchtowers generate ultrasonic field and protect all objects that are strictly inside the towers' convex hull. There is no tower strictly inside the convex hull and no three towers are on a straight line.

The enemy can blow up some towers. If this happens, the protected area is reduced to a convex hull of the remaining towers.



The base commander wants to build headquarters inside the protected area. In order to increase its security, he wants to maximize the number of towers that the enemy needs to blow up to make the headquarters unprotected.

Input

The first line of the input file contains a single integer n ($3 \le n \le 50\,000$) — the number of watchtowers. The next n lines of the input file contain the Cartesian coordinates of watchtowers, one pair of coordinates per line. Coordinates are integer and do not exceed 10^6 by absolute value. Towers are listed in the order of traversal of their convex hull in clockwise direction.

Output

Write to the output file the number of watchtowers the enemy has to blow up to compromise headquarters protection if the headquarters are placed optimally.

Sample input and output

jungle.in	jungle.out
3 0 0 50 50 60 10	1
5 0 0 0 10 10 20 20 10 25 0	2

Problem K. K-Graph Oddity

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

You are given a connected undirected graph with an odd number of vertices. The degree of the vertex, by definition, is the number of edges incident to it. In the given graph the degree of each vertex does not exceed an odd number k. Your task is to color the vertices of this graph into at most k distinct colors, so that the colors of any two adjacent vertices are distinct.

The pictures below show two graphs. The first one has 3 vertices and the second one has 7 vertices. In both graphs degrees of the vertices do not exceed 3 and the vertices are colored into at most 3 different colors marked as ‘ ’, ‘ ’ and ‘ ’.

Input

The first line of the input file contains two integer numbers n and m, where n is the number of vertices in the graph ($3 \le n \le 9999$, n is odd), m is the number of edges in the graph ($2 \le m \le 100\,000$). The following m lines describe edges of the graph, each edge is described by two integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) — the vertex numbers connected by this edge. Each edge is listed at most once. The graph in the input file is connected, so there is a path between any pair of vertices.

Output

On the first line of the output file write a single integer number k — the minimal odd integer number, such that the degree of any vertex does not exceed k. Then write n lines with one integer number c_i ($1 \le c_i \le k$) on a line that denotes the color of i-th vertex.

The colors of any two adjacent vertices must be different. If the graph has multiple different colorings, print any of them. At least one such coloring always exists.

Sample input and output

kgraph.in	kgraph.out
3 2 1 3 3 2	3 1 1 2
7 8 1 4 4 2 2 6 6 3 3 7 4 5 5 6 5 2	3 1 1 1 2 3 2 2