

Problem A. Circle Cover

Input file: *standard input*
Output file: *standard output*
Time limit: 1 second
Memory limit: 256 mebibytes

You are given n points on a plane.

Find a circle that covers all the points and has minimal possible radius.

Input

The first line contains integer n — the number of points ($1 \leq n \leq 100\,000$). The following n lines contain two integers each, coordinates of the points. They do not exceed 10^6 by their absolute values.

Output

The first line of output must contain two floating point numbers, the coordinates of the center of the circle. The second line must contain the radius of the circle. Absolute or relative error must not exceed 10^{-6} .

Examples

standard input	standard output
3	1.00 1.00
0 2	1.4142135624
0 0	
2 0	

Problem B. Perfect Matching

Input file: *standard input*
Output file: *standard output*
Time limit: 1 second
Memory limit: 256 mebibytes

You are given a graph. Find out whether it contains a perfect matching.

Input

The first line of input contains two integers: n and m — the number of vertices and edges in the graph, respectively ($1 \leq n \leq 100$). The following m lines contain two integers each and describe edges of the graph.

Output

Output “YES” or “NO” — whether the graph in the input contains a perfect matching.

Examples

standard input	standard output
6 7 1 2 2 3 1 3 5 6 6 4 4 5 6 2	YES
3 3 1 2 2 3 1 3	NO

Problem C. Global Minimal Cut

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

You are given an undirected weighed graph. Find the cost of the global minimal cut in this graph.

Input

The first line of input contains two integers n and m — the number of vertices and edges in the graph ($2 \leq n \leq 1000$, $1 \leq m \leq 30\,000$). The following m lines describe edges and contain three integers a , b and c each. This triple describes the edge between a and b with cost c ($0 \leq c \leq 10^9$).

Output

Output the cost of the minimal global cut.

Examples

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

Problem D. Matrix Multiplication

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

Little Josh learns how to multiply matrices. He practices by multiplying large binary matrices over \mathbb{F}_2 (the operations in this field are performed modulo 2). Recently he has multiplied two $n \times n$ matrices A and B , and now claims that the result is C .

But Jenny doesn't believe him. She says that he is wrong. Help the children to resolve their dispute. Given matrices A , B and C , check whether indeed $AB = C$.

Input

The first line of the input file contains n — the size of matrices ($1 \leq n \leq 4000$). The following 3 lines describe matrices A , B and C .

Each matrix is described by a line that contains n blocks of $\lceil n/4 \rceil$ hexadecimal digits. If you write the digits in a binary notation, higher bits first, and truncate extra digits in the end, you will get the corresponding row of the matrix. For example, the matrix

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

is described as 28, A8, 68, 78, D0, 88.

Output

Output "YES" if $AB = C$ is true, and "NO" otherwise.

Example

standard input	standard output
6 28 A8 68 78 D0 88 80 40 20 10 08 04 28 A8 68 78 D0 88	YES
6 28 A8 68 78 D0 88 80 40 20 10 08 04 28 B8 68 78 D0 88	NO

Note

Input files in this problem are huge. Please do not submit a solution if you understand that it will not pass because of the time limit — do not create testing jams.

Yes, judges have a solution for this problem in Java that runs well under time limit. Do not use "`std::ifstream`" or "`java.util.Scanner`" to read the input file in this problem.

Problem E. Blind Problem Solving

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 second
 Memory limit: 256 mebibytes

This is an interactive problem. You are to solve a subset sum problem, a variant of a knapsack problem. You are given n items, each weighing w_i grams, and a knapsack with the maximum capacity of c grams. You are to find a subset of these items whose total weight does not exceed c and is maximal possible.

Easy? But you have to do it blindly!

At the beginning you know only two numbers n and c — the number of items and the maximal capacity of a knapsack. There are positive weights w_i and two bit strings inside the jury program: A and B , each having n bits. Each bit string represents a candidate solution to the problem: if the i -th bit is set to 1, then the i -th item is a part of the solution. A stores your previously selected candidate solution. B is a candidate solution that is proposed to you on each turn by flipping a random bit in A . You can either accept or decline this proposal. Your task is to stop when B encodes the optimal solution.

Initially, A is initialized with some value that is fixed for each test case but is not known to you.

At each turn the value of A is copied to B , and then a bit at a uniformly random position of B is flipped. You are given the weight of B which is equal to $\sum_{i=1}^n w_i \cdot B(i)$, where $B(i)$ is the value of i -th bit in B .

As a reply, your program is allowed to perform one of the following three actions:

- **stop** — this is the final action. This means that B encodes the optimal solution. Your program must exit after performing this action.
- **accept** — the value of B is copied to A .
- **decline** — the value of B is thrown out.

After each non-final action next turn starts. You are allowed to perform at most 1000 actions.

Interaction Protocol

Interaction starts with your program reading three integer numbers — the values of n , c and the weight of B for the first turn from the first line of the standard input. Then your program must write its action to the standard output, wait for the jury program to write the weight of the new value of B to the standard input and so on.

Your program must exit after writing the last action (**stop**) to the standard output. Your program must write end-of-line sequence and flush the standard output after each action, including the last one.

It is guaranteed that a jury program chooses the bits to flip randomly, i. e. using a random number generation algorithm with an initial seed fixed for each test case.

Input

The first line of the standard input contains n ($1 \leq n \leq 20$), c ($0 \leq c \leq 10^9$) and the weight of B . It is guaranteed that $1 \leq w_i \leq 10^8$ for any i .

Each of the following lines will contain one integer number — the weight of B at the corresponding turn.

Output

The standard output consists of the actions your program performs. Each action is represented with a single line containing a single word: **stop**, **accept**, or **decline**.

Example

standard input	standard output
2 5 3	decline
2	accept
5	stop

Problem F. Noncommuting Permutations

Input file: *standard input*
Output file: *standard output*
Time limit: 1 second
Memory limit: 256 mebibytes

Permutation is a sequence $\{p_i\}$ of n integers from 1 to n , that contains each integer exactly once.

Product of permutations p and q is a permutation $pq = z$, such that $z_i = p_{q_i}$.

Two permutations p and q *commute*, if $pq = qp$.

For each permutation in input find the permutation that it doesn't commute with.

Input

Input contains description of one or more permutations.

The first line contains an integer n — the number of elements in a permutation ($1 \leq n \leq 100\,000$). The second line contains n integers from 1 to n — the permutation.

The sum of n in one input data doesn't exceed 10^5 .

Output

For every permutation output one or two lines. If there is a permutation that doesn't commute with the given one, output "Yes" at the first line, then output n integers at the second line — any such permutation. In the other case output one line containing a word "No".

Examples

standard input	standard output
3	Yes
2 1 3	2 3 1
2	No
1 2	

Problem G. Detect Shuffling Method

Input file: *standard input*
Output file: *standard output*
Time limit: 1 second
Memory limit: 256 mebibytes

You are working in Laboratory of Artificial Intelligence. Now you are teaching your AI to find difference between two ways of data shuffling. You have decided to use solutions of student teams from Petrozavodsk Training Camps as training data.

Consider a program in Java or C++ that solves some problem from Petrozavodsk Training Camp. Let the whole program be placed into a string s with all characters with ASCII codes less or equal to 32 (space) replaced by spaces. The length of s is between 5 to 20 kilobytes, s is appended with spaces until its length is divisible by 4.

There are two ways to shuffle a string s of length L that you have to distinguish between.

The first way is the following: pick random permutation π of numbers from 1 to L and shuffle characters in s according to π . Let us call such shuffling method “random”.

The second way is the following: divide s to blocks of four consecutive characters. Pick random permutation of numbers from 1 to $L/4$ and shuffle blocks according to it. Then for each block pick random permutation of numbers from 1 to 4 and shuffle characters of this block with it. Random permutations for each block are selected independently. After that $\lfloor L/10 \rfloor$ times choose two integers from 1 to L at random and swap characters at corresponding positions. Let us call such shuffling method “block”.

You are given a file with n lines, each one is either random or block shuffle of the same string s , exactly half of these lines are produced with random shuffle, the other half are produced with block shuffle. For each line identify whether it was produced with random or block shuffle. You must detect shuffling method correctly for at least 80% of lines in each test case.

Input

The first line of each input file contains n — the number of lines to follow ($10 \leq n \leq 100$, n is even).

Each of the following lines has the same length L from 5000 to 20000 and contains characters with ASCII codes from 32 to 126. Each of them was produced from the same string s with either random shuffle or block shuffle. Exactly $n/2$ of these lines were produced with random shuffle, another $n/2$ lines were produced with block shuffle. The string s is source code of some submitted solution from Petrozavodsk Training Camp written either in C++ or in Java with all characters with codes less than 32 replaced with spaces.

Sample input corresponds to “Hello World” program in C++, it violates the restriction that all lines have length between 5000 and 20000, any correctly formatted answer to this input would be accepted (note: judges’ program correctly solves this input as well, however).

Output

For each line print “random” if this line represents randomly shuffled s , or “block” if the line represents s shuffled with block shuffle.

Sample output in the problem statement is the correct answer to the sample input.

Examples

standard input
<pre> 10 c<({ ldulm ca}; p itno, loerWdd"!oi t">fsntrielH) % ", si#inn)(" ! dtn"or #iis np "Woo ,%)lfn) {clr m>t(ud; tl<idsec la, H} (i "ne"i l s,tdio(n)p l { };ud ,tirnlroWc <slH"etn # o ema %inic " >f"(i"!>d i e nirr ai)pi(; d cnt ,l o s"n {"W,l>!" m Htt <o }("uil#s %)nd c fodle H ";niorWloidtldue"!>d { n ()le" > plm ,t,rn aio s"ini#cs <c(%ft } lWcu# eniaf(eo dn !>) "dtisd oni %n tr l" lt,i)mc H(; < } p" s,"r { ilo idtollrWa> n <cs m o " s,ledur it itn%"f(i#cnlH el ,o)"d! "{ } (n ; } pi lttdnri an" l# "ms o p>{i i ct r %o)osi ,nfHni!d "u;l (l cd" W)e(<),e iencsc< !d ""ileW{ olo rldu# };) n (s ,"tin "(f% ptHrntoidl), ami > nnod)plo{c #" n%(elcu ", t d t,s") (iims nWl ;iiadt! r rf }i H> e "o<l </pre>
standard output
<pre> block random block random block random block random block random </pre>

Problem H. Entropy

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 second
 Memory limit: 256 mebibytes

In 1948 Claude E. Shannon in “The Mathematical Theory of Communication” has introduced his famous formula for the entropy of a discrete set of probabilities p_1, \dots, p_n :

$$H = - \sum p_i \log_2 p_i .$$

We will apply this formula to an arbitrary text string by letting p_i be the relative frequencies of occurrence of characters in the string. For example, the entropy of the string “Northeastern European Regional Contest” with the length of 38 characters (including 3 spaces) is 3.883 with 3 digits after decimal point. The following table shows relative frequencies and the corresponding summands for the entropy of this string.

char	occurs	p_i	$-p_i \log_2 p_i$	char	occurs	p_i	$-p_i \log_2 p_i$
space	3	0.079	0.289	i	1	0.026	0.138
C	1	0.026	0.138	l	1	0.026	0.138
E	1	0.026	0.138	n	4	0.105	0.342
N	1	0.026	0.138	o	4	0.105	0.342
R	1	0.026	0.138	p	1	0.026	0.138
a	3	0.079	0.289	r	3	0.079	0.289
e	5	0.132	0.385	s	2	0.053	0.224
g	1	0.026	0.138	t	4	0.105	0.342
h	1	0.026	0.138	u	1	0.026	0.138

Your task is to find a string with the given entropy.

Input

Input file consists of a single real number H ($0.00 \leq H \leq 6.00$) with 2 digits after decimal point.

Output

Write to the output file a line with a single string of at least one and up to 1000 characters ‘0’–‘9’, ‘a’–‘z’, ‘A’–‘Z’, ‘.’ (dot), and spaces. This string must have the entropy within 0.005 of H .

Example

standard input	standard output
3.88	Northeastern European Regional Contest

Problem I. Frequent Permutations

Input file: *standard input*
Output file: *standard output*
Time limit: 4 seconds
Memory limit: 256 mebibytes

Peter is developing new casino software. Now he is writing a part of software that will shuffle the cards deck. To do so he needs to generate random permutation of integers from 1 to n .

Unfortunately, Peter doesn't know good algorithm to do so. So he uses the following algorithm. First initialize $a[i] = i$. Then t times choose uniformly random i from 1 to n and random j from 1 to n and swap $a[i]$ and $a[j]$.

John learned about Peter's method to generate a random permutation and decided to use his knowledge to win in the casino. John wonders what permutation occurs most frequently among permutations generated by Peter's method and what is its probability.

Help John to find that out.

Input

The input file contains several test cases.

Each test case consists of two integers on a line: n and t ($1 \leq n \leq 14$, $0 \leq t \leq 300$).

The last test case is followed by two zeroes that must not be processed.

Output

For each test case output two lines. The first line must contain n integers: the permutation that occurs most frequently. If there are several such permutations, output lexicographically smallest one. The second line must contain the probability of generating this permutation as an irreducible fraction p/q .

Examples

standard input	standard output
3 3	1 2 3
3 4	5/27
0 0	1 2 3
	43/243

Problem J. Genome Research

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 second
 Memory limit: 256 mebibytes

Scientists of Retrozavodsk Laboratory of Genome Research are preparing to analyze species from Pandora — a new 3D planet discovered at the nearby star. They have decided to start by analyzing genotypes of small bacteria called *E-lectus*.

After decoding its genome, the scientists have learned that there are m different genotypes of *E-lectus*. *E-lectus* lives in colonies and bacteria of various genotype are distributed in some proportion in each colony. This proportion is approximately the same in each colony. One of genotypes, known as *unific* is dominating in each colony having more species than any other genotype.

Scientists decided to take several samples from each of the n colonies. Their funds are limited, so they hired a chinese company *Tzen Triz Bir Co* headed by Prof Chu Ro. The company took samples from each colony by sequencing genome of 50 to 1000 bacteria from it. However, Chu Ro has his own interest in analyzing Pandora nature, so he decided to spoil Retrozavodsk Lab study. So he ordered to make frauds of two types to the samples.

First they have chosen some colonies and added some bacteria with *unific* genotype to samples from these colonies.

Secondly they have chosen some other colonies and replaced some bacteria from these colonies that have genotype different from *unific* to species with *unific* genotype. Only bacteria with genotype that have at most 15% species in a total population could be replaced.

The resulting distorted distributions of genotypes among colonies were presented to Retrozavodsk Scientists. Fortunately, scientists are not idiots, so they understood that something is wrong with the data. Now they are trying to recover, data from which colonies were not frauded (type 0), data from which colonies were changed by adding some *unific* bacteria (type 1) and data from which colonies were changed by replacing some bacteria with *unific* bacteria (type 2).

You are given distributions as input. For each colony you have to detect its type.

Note

In this problem test generation and evaluation are different from traditional.

Each test case was generated using the following algorithm. First, integer n was selected between 200 and 1000 inclusive, and integer m is selected between 5 and 10 inclusive, index u of *unific* genotype is selected between 1 and m , inclusive. Distribution of genotypes $P = (p_1, p_2, \dots, p_m)$ is selected ($0.01 \leq p_i \leq 0.50$, $p_1 + p_2 + \dots + p_m = 1$, $p_u > p_i$ for all $i \neq u$). There is at least one other genotype which has $p_i > 0.15$ and at least two genotypes which have $p_i < 0.15$, sum of probabilities of all genotypes that have $p_i < 0.15$ is at least 0.20.

For each of n colonies number of sampled species k_i is selected uniformly at random between 50 and 1000. After that k_i times a species genotype is generated with distribution P . Denote number of species with corresponding genotypes as $a_{i,1}, a_{i,2}, \dots, a_{i,m}$.

Next two probabilities r_1 and r_2 are selected between 0.25 and 0.35 each. All n colonies are considered one after another. Each colony is left normal with probability $1 - r_1 - r_2$, is frauded by adding *unific* species with probability r_1 and is frauded by replacing some species by *unific* ones with probability r_2 .

Fraud by adding *unific* species is performed in the following way. Let k_i samples be taken from the corresponding colony. An integer f_i is generated uniformly at random between $0.25k_i$ and $1.25k_i$, inclusive. Then f_i *unific* genotypes are added to the data ($a_{i,u} += f_i$).

Fraud by replacing some species with *unific* is performed in the following way. For each genotype such that $p_j \leq 0.15$ an integer $f_{i,j}$ is selected between $0.75a_{i,j}$ and $a_{i,j}$, inclusive. Then $f_{i,j}$ species with genotype j are replaced by species with *unific* genotype ($a_{i,j} -= f_{i,j}$, $a_{i,u} += f_{i,j}$).

After that percentages of bacteria of corresponding genotype in each colony are rounded to exactly 2 digits after the decimal point. Name these percentages as $b_{i,j}$. They are provided in the input file.

Your output will be accepted if you correctly identify at least 2/3 of colonies of each of types 0, 1 and 2.

Sample input has $n = 45$ to fit the page. Any correctly formatted output for sample input (test 1) will be accepted. It was generated using rules above with $n = 45$, $m = 7$, $P = (0.15, 0.10, 0.02, 0.25, 0.10, 0.35, 0.03)$, $r_1 = r_2 = 0.30$.

Judges' program for general case correctly solves samples input without any additional suggestions, making only

two incorrect identifications.

Input

The first line of the input file contains n and m ($200 \leq n \leq 1000$, $5 \leq m \leq 10$, except sample input which has $n = 45$).

The following n lines contain m real numbers given with exactly two digits after the decimal point — $b_{i,j}$. Due to rounding errors they may not sum exactly to 100.

Output

For each colony output its type — 0, 1 or 2.

Examples

standard input	standard output
45 7	
10.67 7.11 0.91 16.71 6.04 57.24 1.32	1
10.08 6.79 1.71 16.26 5.90 56.86 2.40	1
3.70 2.47 1.23 25.93 2.47 64.20 0.00	2
14.31 9.19 1.94 28.27 8.13 36.04 2.12	0
3.23 1.08 1.08 29.03 2.15 63.44 0.00	2
2.36 0.79 0.52 25.72 0.52 69.82 0.26	2
13.75 9.82 1.96 24.75 11.98 34.18 3.54	0
1.19 0.00 0.40 21.43 1.98 74.21 0.79	2
14.78 7.26 2.69 26.61 10.75 35.22 2.69	0
0.54 0.98 0.22 24.21 0.11 73.62 0.33	2
17.20 9.47 1.87 21.73 9.87 36.67 3.20	0
10.06 12.43 2.37 23.67 12.43 37.87 1.18	0
13.78 9.34 1.68 25.11 9.80 37.37 2.91	0
0.85 0.71 0.28 24.96 0.56 72.50 0.14	2
18.02 8.71 2.10 26.43 9.91 32.43 2.40	0
15.96 9.64 2.01 22.69 9.24 36.95 3.51	0
3.07 2.50 0.58 25.72 0.77 67.18 0.19	2
1.33 0.88 0.11 24.34 2.21 71.02 0.11	2
16.25 9.05 2.84 25.63 9.71 33.15 3.38	0
12.70 7.94 0.79 24.60 15.08 32.54 6.35	0
14.73 9.45 0.18 25.45 7.82 37.82 4.55	0
0.35 1.39 0.69 25.00 0.35 72.22 0.00	2
8.02 5.73 1.53 17.18 6.49 60.31 0.76	1
13.99 7.77 2.07 23.83 11.40 37.65 3.28	0
3.05 0.55 0.14 26.87 0.83 68.01 0.55	2
13.35 10.96 0.80 26.69 12.15 33.86 2.19	0
13.33 15.24 2.54 21.27 9.21 34.92 3.49	0
7.63 5.10 0.82 12.46 5.21 67.51 1.26	1
16.42 10.22 1.82 22.63 12.04 33.39 3.47	0
3.51 1.00 0.75 27.32 0.75 66.67 0.00	2
14.29 9.52 1.86 25.05 10.35 35.20 3.73	0
11.65 4.82 2.01 18.88 7.63 52.61 2.41	1
13.16 11.28 3.01 24.44 12.03 32.33 3.76	0
0.13 0.78 0.39 23.28 1.17 73.99 0.26	2
0.51 1.03 1.03 27.18 1.03 68.72 0.51	2
14.89 9.24 1.52 25.22 10.87 33.70 4.57	0
17.46 9.66 3.22 23.90 8.98 34.75 2.03	0
13.91 13.53 1.88 24.25 8.08 34.77 3.57	0
14.40 9.89 1.65 25.16 9.56 36.37 2.97	0
15.88 12.42 2.57 25.28 10.40 30.87 2.57	0
1.94 1.77 0.00 28.87 0.97 66.29 0.16	2
14.07 8.89 2.22 22.22 17.04 31.11 4.44	0
6.45 3.90 0.78 12.48 4.26 70.78 1.35	1
15.13 8.79 1.02 23.31 10.43 37.42 3.89	0
15.21 10.84 2.27 21.68 11.49 34.63 3.88	0

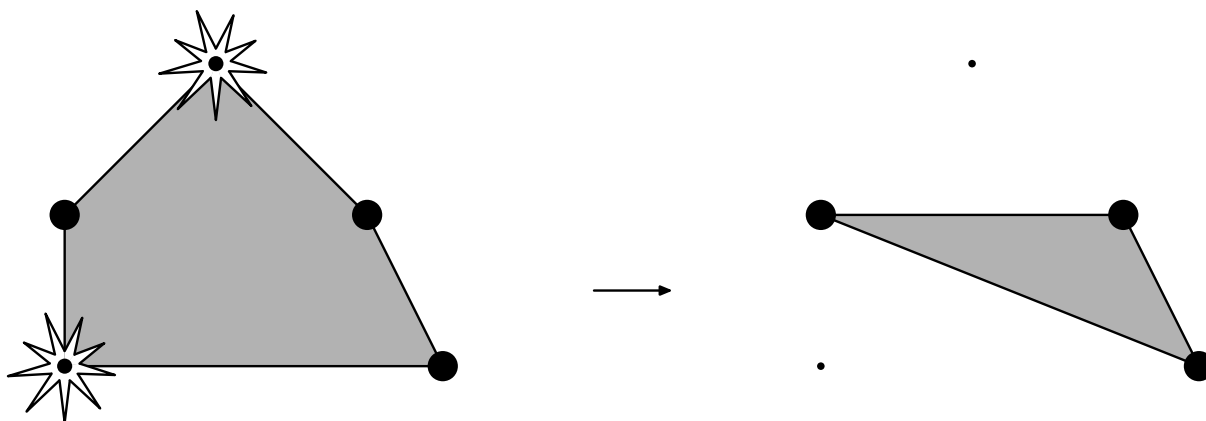
Problem K. Jungle Outpost

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

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 ($3 \leq n \leq 50\,000$) — 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.

Example

standard input	standard output
3 0 0 50 50 60 10	1
5 0 0 0 10 10 20 20 10 25 0	2

Problem L. LED-led Paths

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 second
 Memory limit: 256 mebibytes

In a large city there are n junctions and m one-way streets connecting those junctions. It is known that one can't walk along the streets infinitely, as there is no directed cycle of streets in the city.

The tourism department has decided to install colored illumination using the LED (light-emitting diode) technology. Each street is to be illuminated with either red (R), green (G), or blue (B) color.

The official city maps for tourists will indicate the selected colors. All continuous paths illuminated by the same color will be suggested for walking and sightseeing. As LEDs will lead people along beautiful ways, these paths will be called LED-led paths.

The health department says that if there is a very long single-color LED-led path, some tourists might overestimate their strength, walk for too long, get too tired, and dislike the city.

Propose a three-color illumination plan in which no single-color LED-led path consists of more than 42 streets.

Input

The first line of the input contains two integers n and m — the number of junctions and streets, respectively ($2 \leq n \leq 50\,000$; $1 \leq m \leq 200\,000$).

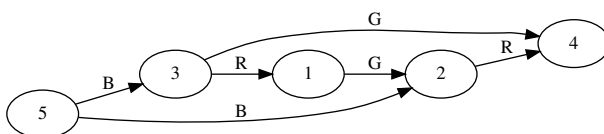
Each of the following m lines contains two integers u_i and v_i , denoting a one-way street from junction u_i to junction v_i ($1 \leq u_i, v_i \leq n$; $u_i \neq v_i$).

The given city is guaranteed to be acyclic. Each pair of junctions is connected by at most one street.

Output

For each street, in the order of input, output its color on a separate line — a single letter R, G, or B.

Example

standard input	standard output	Illustration
5 6 5 3 3 1 1 2 2 4 5 2 3 4	B R G R B G G	

Note

In this example, all single-color LED-led paths are not longer than one street (and therefore not longer than 42 streets), which is absolutely OK according to the health department.