icpc international collegiate programming contest

ICPC Asia West Regionals 2019

ICPC Asia West Continent

Final Contest

Official Problem Set

icpc international collegiate programming contest

icpc.foundation

icpc global programming tools sponsor

15th January 2020

You get 28 Pages, 13 Problems & 300 Minutes

This page is left intentionally blank.

Problem A: Q-SQL

Ahmad and Nabi just learned about SQL (Structured Query Language) in their database course. Because they love programming, they want to develop a small DBMS (Database Management System) as a fun project. They want to create a program that can run a very simple SQL SELECT query against a CSV (Comma Separated Values) file. For the first version of their program, they want to implement the functionality that supports the following SQL syntax:

```
        SELECT columns
        WHERE expression;
        -------------------------------------------------
        columns: * | column[,column...]
        expression: column="value" | (column="value") | column="value" {AND |
OR} expression | (column="value" {AND | OR} expression)
```

Note:

- There is no FROM clause; it is not supported as we are only reading data from a single file.
- WHERE clause is **optional**. eg: SELECT *; is a valid input.
- Only EQUALITY, AND, OR keywords/operators are supported, but they can be nested.
- In WHERE clause, column value is **always** wrapped in "double quotations". There won't be any whitespaces inside double quotations for column values.
- The SELECT statement and Equality operation are both **case insensitive**.
- Expressions are evaluated from left-to-right, and precedence is specified by parentheses.

Input

There will be only one case per input file. The first few lines contains the SQL select query, always ending with a **semicolon (;)**. The length of the query including white spaces and the semicolon will never be more than **666** characters. The select query can contain any **valid ASCII characters** but will follow the syntax specified above.

The next line contains **L**, the number of lines in the CSV file, **2 ≤ L ≤ 100000.** Then **L** lines will follow representing the CSV file. The first line will contain column names which will all be unique. Each of the next **L - 1** lines will contain the CSV data.

Column name and column value will only contain alphanumeric characters **(a-z, A-Z, 0-9)**. Neither column name or value will contain any of the special keywords like SELECT, FROM, WHERE, AND, OR, etc.

1 ≤ num of column, len(column name), len(column value) **≤ 100**

Note that the input file can be huge, please use faster I/O methods. However it is guaranteed that no input file will exceed 18 MB.

Output

For each query, first output the column names. For select * queries, print all the column names of the CSV as given in the input, **preserving their order and case**. For queries where the columns to be selected are given, print the selected columns **preserving the order and case specified in the select query**. The order and case of columns in the query can be different than that in the CSV file. Lastly output all rows of the CSV file matching the query. The rows should be printed **in the order given in the input**, that is if row_1 and row_3 match, row_1 should be printed first followed by row_3. The column values for each row will however follow the **order of the column names printed in the last step, but the case of them must be preserved as in the CSV file**.

Note:
The query may have extra spaces. Note the case insensitiveness of the keywords, names and values too:
SelEct   a,   B, c where ((((A = "A"   AnD b=   "2"))));

Sample Input

```
select a,   D  where A="1" and
(     b="1" or d="1");
13
a,b,c,d
1,1,1,1
1,1,1,2
1,1,1,1
1,1,2,2
1,1,2,1
1,1,2,2
1,2,1,1
1,2,1,2
1,2,1,1
1,2,2,2
1,2,2,1
1,2,2,2
```

Output for Sample Input

```
a,D
1,1
1,2
1,1
1,2
1,1
1,2
1,1
1,1
1,1
```

Explanation:

Result	a	b	c	d
■	1	1	1	1
■	1	1	1	2
■	1	1	1	1
■	1	1	2	2
■	1	1	2	1
■	1	1	2	2
■	1	2	1	1
	1	2	1	2
■	1	2	1	1
	1	2	2	2
■	1	2	2	1
	1	2	2	2
a="1" and (b="1" or d="1")				

Problem B: Auxiliary Prime

A number is said to be auxiliary prime if it can be changed into a prime number by changing at most one of its digits. Note that you are not allowed to change the leading digit of the number to zero.

Given an integer **N**, check whether the number is auxiliary prime or not?

Input

The first line of the input contains an integer **T** $(1 \le T \le 10^6)$, denoting the number of test cases. Then the **T** test cases follow. For each case, the only line of input contains an integer **N** $(1 \le N \le 10^6)$.

Output

For each test case, output in a single line "**yes**" or "**no**" (without quotes) depending on whether the number is auxiliary prime or not.

Sample Input

```
4
17
1003
200
10030
```

Output for Sample Input

```
yes
yes
no
yes
```

Explanation

- In the first test case, 17 is already a prime. Thus, it's an auxiliary prime.
- In the second test case, you can change the third digit (from the left) to 9 and get the number 1093, which is a prime. Thus, 1003 is an auxiliary prime.
- In the third test case, you can't change 200 to a prime number by changing at most one of the digits. So it's not an auxiliary prime.

NB: Large dataset, use faster I/O.

This page is left intentionally blank.

Problem C: Currently at Top

The website Codeforces has a recent action window. This window stores n active blog posts.

Every minute, a user visits the website and :

- With probability $p = \frac{s}{t}$, he chooses one of the n blog posts from the window randomly, and comments on it. On doing so, this post comes on the top. So if the posts from top to bottom were $A_1, A_2, \ldots, A_n$ and he comments on A_i, the new order of posts becomes $A_i, A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$.
- With probability $1-p$, he posts a new blog post which appears on the top and removes the bottom most post from the window. That is, if the window looked like $A_1, A_2, \ldots, A_n$ from top to bottom and he posts a blog post B, it would look like $B, A_1, A_2, \ldots, A_{n-1}$ after the post.

Find the expected number of minutes after which the post currently at top is kicked out of the window.

The probability p is given as a rational number $\frac{s}{t}$. The answer can be written as $\frac{A}{B}$, where A and B are positive coprime integers. Print the value of AB^{-1} modulo 10^9+7, where B^{-1} denotes the modular inverse of B modulo 10^9+7.

Input

First line will contain a single integer **C (C < 20)**, denoting the number of test cases. Each of the next lines contains three integers, **n ($1 \le n \le 10^9$), s , t ($0 \le s < t < 10^9 + 7$, $t \ne 0$)** for each case.

Output

For each case, print the required expected value modulo **10^9+7** in a single line.

Sample Input	Output for Sample Input

2
1 1 2
4 127749536 137669261
	2
	217930292

Explanation

In sample input # 1, there is only one blog post. It will be kicked out when a new blog post appears. The probability of succeeding is $\frac{1}{2}$ in every minute. So, the expected number of minutes is 2.

In sample input #2, if the current posts are $1, 2, 3, 4$ in order from top to bottom, then the window after one minute looks like:

- 1, 2, 3, 4 with probability $\frac{p}{n}$ (comment on 1)
- 2, 1, 3, 4 with probability $\frac{p}{n}$ (comment on 2)
- 3, 1, 2, 4 with probability $\frac{p}{n}$ (comment on 3)
- 4 ,1, 2, 3 with probability $\frac{p}{n}$ (comment on 4)
- 5, 1, 2, 3 with probability $1-p$ (new blog post 5)

We have to find the expected amount of time after which 1 is kicked out of the window. For example, one possibility is: (1, 2, 3, 4) -> (3, 1, 2, 4) -> (5, 3, 1, 2) -> (1, 5, 3, 2) -> (6, 1, 5, 3) -> (3, 6, 1, 5) -> (5, 3, 6, 1) -> (7, 5, 3, 6) taking 7 minutes.

This page is left intentionally blank.

Problem D: Shortest Path with Discounts

Townsville city contains **N** junctions. There are **M** directed roads which connect the junctions. Alice, a computer programmer, lives at junction **S**. She travels to her workplace, which is at junction **T**. Whenever Alice uses a road, she needs to pay some cost in rubles which is fixed for each road. Her company has given her **K** discount coupons. The i-th coupon provides a discount of $\mathbf{d_i}$ rubles. Each discount can be used at most once and for at most one road. Help Alice find the cheapest route from junction **S** to junction **T**.

Note:
- A road can be used by Alice at most once in her journey.
- There is at most 1 directed road from one junction to another.

Input

The first line of the input contains **C** $(1 \le C \le 20)$, number of test cases to follow.
The first line of each test case contains three space-separated integers **N** $(1\le N \le 300)$, **M** $(1 \le M \le 10^4)$ and **K** $(1 \le K \le 10^4)$, representing the number of junctions, number of roads and number of discount coupons respectively. The second line contains two space-separated integers, **S** and **T** $(1 \le S, T \le N)$. The third line contains **K** space-separated integers, where the i-th integer represents $\mathbf{d_i}$ $(1 \le d_i \le 1000)$, the i-th discount coupon. Each of the next **M** lines contain three space-separated integers, **u, v** $(1 \le u, v \le N)$, **w** $(1000 \le w \le 10^9)$, which represents a directed road from **u** to **v** with cost **w**.

Output

For each test case, print a single line containing an integer, which is the minimum cost that Alice needs to pay to go from **S** to **T**. If no path exists from **S** to **T**, print **−1**.

Sample Input

```
2
3 4 1
1 3
10
1 2 1010
2 3 2000
1 3 1500
1 1 1100
3 1 1
1 3
10
1 2 1000
```

Output for Sample Input

```
1490
-1
```

Explanation

For the first test case, one possible path is 1−>2−>3 and the discount coupon can be applied on the road (1,2) to get a cost of (1010−10)+2000=3000 rubles. However, the optimal path would be to take the road (1,3) and use the discount coupon there and pay 1500−10=1490 rubles.

This page is left intentionally blank.

Problem E: Make GCD

The gcd of a non-empty array is defined as the greatest common divisor of the values in the array. Define the beauty of an array as the sum of gcd of all of its subarrays. For example, the beauty of array [4, 6, 3] is 4 + 6 + 3 + gcd(4, 6) + gcd(6, 3) + gcd(4, 6, 3) = 4 + 6 + 3 + 2 + 3 + 1 = 19.

Given an integer **X**, find a non empty array of length less than or equal to **10^5** with the values in the range of **[1, 10^5]** such that the beauty of the array is equal to **X**, or claim that such an array doesn't exist.

Input

First line of the input file contains the number of test cases, **T ($1 \le T \le 30$)**. Following T line contains a single integer, **X ($1 \le X \le 10^{13}$)**.

Output

For each test case, if there is no non empty array of length less than or equal to **10^5** with the values in the range **[1, 10^5]** which has a beauty value of **X**, print NO on a single line. Else, find such an array, say **A_1, A_2, ..., A_n** :

- Print YES on the first line.
- Print **n** on the second line.
- Print the space separated array **A_1, A_2, ..., A_n** on the third line.

If there are more than one possible valid outputs, output any.

Sample Input

```
2
19
19
```

Output for Sample Input

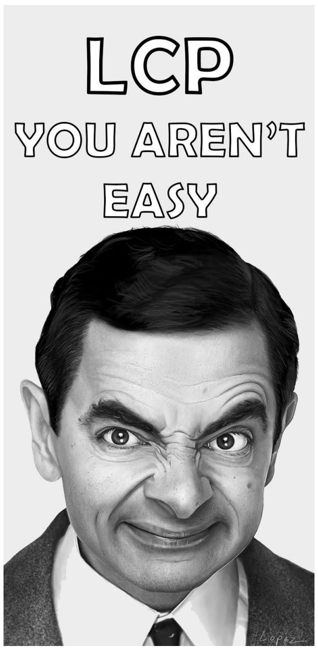
```
YES
3
4 6 3
YES
1
19
```

This page is left intentionally blank.

Problem F: Finding LCP is not always easy

Mr. Bean is studying on string-processing algorithms. Recently he learnt about the term: Longest common prefix which is called LCP in short form. From one website, he got to know that LCP plays an important role in different string related data structures. It helps to easily find out which string can have less rank lexicographically. Sorting a list of string also depends on the LCP of two strings.

Mr. Bean's lovely teddy bear Beanie is really upset for the last few days. Because Mr. Bean wasn't giving enough time to Beanie as he was very busy with learning about LCP. Being angry, Beanie then asked Mr. Bean to solve a problem (though Beanie has no idea how to solve it).



Initially, Mr. Bean will be given a string **S** containing lowercase letters. You will have to perform the following operations on it:

- Insert a string of length **M** into **S** after the position **P** where each character of this new string will be **X**.
- Delete **N** characters from **S** starting from the position **P**. At any stage, the string length will be always greater than **N**.
- Reverse a substring of **S** where the substring starts from position **P1** to **P2**.
- Calculate the LCP of two substrings of **S**, one substring will start from position **P1** to **P2**, and another will start from **Q1** to **Q2**. These two substrings may overlap.

Mr. Bean is trying to figure out a solution to the problem for a long time. But he can't find any. Suddenly one idea came to his mind. There should be one or more people in his fan base who are very talented programmers. From a source, he got to know that you guys are waiting for problems so that you can solve. So, he is asking you for help. Can you help him solve the problem?

Input

The first line of the input contains a single integer **T** denoting the number of test cases.
Each case starts with a line containing the string **S**. Next line contains a single integer **Q** describing the number of operations need to perform on the string. Each of the next **Q** lines will contain any one type of the operations in the following format:

1 M P X: The insert operation
2 N P: The delete operation
3 P1 P2: The reverse operation
4 P1 P2 Q1 Q2: The calculation of LCP.

N.B.: Input file is huge. Please use faster I/O.

Constraints

- **$1 \le T \le 5$**
- **$1 \le |S| \le 50{,}000$**
- **$1 \le Q, M, N \le 50{,}000$**
- **$1 \le P \le$ Length of S at that time**
- **$1 \le P1 \le P2 \le$ Length of S at that time**
- **$1 \le Q1 \le Q2 \le$ Length of S at that time**
- **The length of the string will never cross the limit of 10^6 at any stage**

Output

For each test case, the first line should contain the case number in the format "**Case Z:**" (without quotes) where **Z** is the case number. After that for operation type of **4**, you need to print the answer.

Sample Input

```
1
abcde
5
1 1 4 a
4 1 2 5 6
1 1 4 b
3 5 6
4 1 2 5 6
```

Output for Sample Input

```
Case 1:
1
2
```

Problem G: Fire in the Grid

FG is a board game played on **N** by **M** board (**N** rows and **M** columns). The left uppermost cell is numbered as $(1,1)$ and the right lowermost cell is numbered as (N, M). The rules of the game is as followed.

1) Some of the cells contain bombs. These bombs are of two colors: Red and Green.
2) If a red bomb is fired, it fires all the bombs in its horizontal path. That is, a red bomb at cell (x,y) explodes all the bombs that belongs to that row **x**.
3) If a green bomb is fired at cell (x, y), it fires all the bombs which has column value **y**.

In the given figure you can see a 6 x 6 grid.
It has two red bombs at cell (2,1), (4,3) and two green bombs at cell (2,5) and (4,1).
There can be a series of bombing depending on the bomb we choose to fire.

Fire bomb at cell (2,1): It will fire bomb at (2,5). So, in total two bombs will be fired.
Fire bomb at cell (2,5): It will not fire anything else. So, only one bomb will be fired.
Fire bomb at cell (4,1): It will fire bomb at (2,1) and bomb at (2,1) will fire bomb at (2,5). So, in total three bombs will be fired.
Fire bomb at cell (4,3): All the four bombs will be fired.

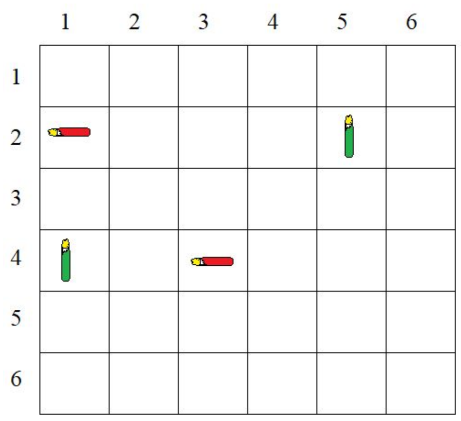


Figure: FG Game Board

In this problem, you are given the current state of the game. You have to decide the maximum number of bombs that can be fired by a single move.

Input

Input begins with an integer **T (1 ≤ T ≤ 10)** denoting the number of test cases. Each of the test cases starts with three integers **N, M** and **B (1 ≤ N, M, B ≤ 40000)** denoting the number of rows, columns and bombs. Each of the next **B** lines contains three integers **x, y** and **C** which denote there is a bomb of color **C** at **x-th** row and **y-th** column. **C = 0** means its a red bomb and **C = 1** means it is a green bomb.

Output

For each test case print, the case number followed by the maximum number of bombs that can be fired by a single move.

Sample Input

```
1
5 5 4
2 1 0
2 5 1
4 1 1
4 3 0
```

Output for Sample Input

```
Case 1: 4
```

NB: Large dataset, use faster I/O.

This page is left intentionally blank.

Problem H: Sarfaranga Rally

Sarfaranga is a desert in the northern areas of Pakistan. It is a major town in the Baltistan region which abounds in natural beauty. Located at the foothills of the mighty Karakoram Range, it is often the last town where mountaineers gather to give final touches to their K2 (the second highest peak in the world) expedition. Sarfaranga is located just outside of Skardu and every year a jeep rally event is organized there to see the best drivers and cars that can cross the desert. Many drivers from far and wide participate in this rally both for passion and for the prestige.

The rally has been on-going for several years with familiar faces and, many times, familiar winners. This year its organizers thought of a way to “spice things up” a bit. They thought that in addition to seeing who’s got the best car and driving skills, they will also see who the smartest driver is. To achieve that, they came up with an interesting plan. The rally has been divided into **N** stages each consisting of **M** blocks like a 2D grid fashion. Stages are numbered from **1 to N** (top to bottom) and blocks of each stage are numbered from **1 to M** (left to right). Each block has an entry/exit gate to an adjacent block such that a driver can only move to a top, bottom, left, or right block from his/her current block position. Each block has a penalty to cross through that block. All drivers must start from the **1st block of the 1st stage** and end up in the **Mth block of Nth stage**. Note that, drivers can not move out of the grid as it will cause immediate disqualification.

You are a rally driver in the Sarfaranga desert rally. Just before the start of the race, each driver is provided with a map of the rally stage that includes the penalty associated with each block. You job is to quickly figure out the path that will result in the least penalty to win the race.

Input

The first line in the input file is an integer **T (1 ≤ T ≤ 50)** denoting the number of test cases. Each test case starts with two integers **N and M (1 ≤ N, M ≤ 20)**. Next line of the test case contains stage-wise entry of the penalty of each block **(0 ≤ Penalty of each block ≤ 200)**.

Output

For each test case, print a single line with the case number followed by the minimal penalty incurred if we follow the optimal path. See samples below for more clarification.

Sample Input

```
2
3 3
1 2 3 5 4 3 1 0 5
4 4
1 1 1 0 4 5 6 0 4 4 4 0 5 7 8 1
```

Output for Sample Input

```
Case 1: Penalty = 12
Case 2: Penalty = 4
```

This page is left intentionally blank.

Problem I: Subarray Optimization

You are given a sequence **P** of **n** lowercase English letters.
For any subarray (i, j), i.e. $P_i, P_{i+1}, \ldots, P_{j-1}, P_j$, we define:

Min(i,j) = *minimum element of the subarray*

Distinct(i,j) = *number of distinct elements in the subarray*

You need to calculate the following expression:

$$\sum_{i=1}^{n} \sum_{j=i}^{n} Min(i,j) \times Distinct(i,j)$$

Input

The first line of the input contains **t (1 ≤ t ≤ 5)** - the number of test cases.
Each test case is described as follows:

- The first line contains **n (1 ≤ n ≤ 10^5)** - the length of the sequence **P**.
- The second line contains **n** space separated integers, **P_i (1 ≤ P_i ≤ 10^5)** - denoting the sequence.

Output

For each test case, output in a single line the value of the expression in the statement.
It can be guaranteed that the answer will always fit in a 64-bit signed integer.

Sample Input

```
3
3
1 2 3
3
1 2 1
3
1 2 2
```

Output for Sample Input

```
15
10
11
```

This page is left intentionally blank.

Problem J: How Many Solutions?

Given the value of integer **N** how many solutions does the following equation have?

$$\frac{1}{x}+\frac{1}{y}=\frac{1}{N}$$

If **x** and **y** are integers there is only a finite number of solutions but if **x** and **y** are real numbers then there can be an infinite number of solutions. What if **x** and **y** are floating-point numbers with limited size, e.g. **x** and **y** are floating point numbers with **d** digits after the decimal points, how many different solutions will be there?

Input

Input file contains at most **2000** lines of input. Each line contains two integers **N (0 < N ≤ 10000000000)** and **d (0 ≤ d ≤ 1000)**, here **d** means that there can be maximum **d** digits after the decimal point. Input is terminated by a line containing two zeros. This line should not be processed.

Output

For each line of input, produce one line of output which contains an integer **T**. This line contains the number of different solutions the equation has for the given value of **N** and **d**. As the value of **T** can be very large so output **T** modulo **1000007**.

Sample Input	Output for Sample Input

23 10
10 2
0 0
	2645
	97

This page is left intentionally blank.

Problem K: Base Stations

T-net, a cellular network company, has established **n** base stations at **n** distinct positions, and assigned a frequency to each base station. Each base station covers clients within a certain distance. In general, the coverage area of base stations may overlap. This may lead to interference of the signals for a client who is in the coverage area of more than one base station. Thus, for a client within the coverage area of at least one base station, there must be a base station with a unique frequency among base stations covering the client. So, the client can connect to the network without interference. A point **p** is called a covered place if it is covered by at least one base station. A covered place **p** is called conflict-free if among base stations covering **p**, there is a base station with a unique frequency (i.e. there exists a base station whose frequency is different from the frequency of the other base stations covering **p**). Indeed, a client standing at a conflict-free covered place can connect to the network without interference. Two covered places **p** and **q** are called equivalent if all points on the segment **pq** (including **p** and **q**) are covered by the same subset of base stations. Otherwise, they are called non-equivalent. Now, T-net is wondering maximum how many covered places exist in the coverage area of its network that are not conflict-free and are pairwise non-equivalent.

For your ease, assume that the world is one dimensional. So, the coverage area of each base station can be modeled by a closed interval **[a, b]**, where **a** and **b** are two distinct numbers. Also, assume that the frequency assigned to each base station is a positive integer.

Input

There will be multiple test cases. You must take input until end of file. For each case, the first line of the input contains an integer **n** denoting the number of base stations **($1 \le n \le 10^5$)**. Each of the next **n** lines describes a base station by three positive integers **a**, **b**, and **f**, where **[a, b]** denotes the coverage area of the base station, and **f** denotes its frequency **($1 \le a < b \le 10^5$** and **$1 \le f \le 10^5$)**. **I/O file is huge, please use faster I/O.**

Output

For each case, print the number of pairwise non-equivalent covered places that are not conflict-free in a single line.

Sample Input

```
3
1 5 1
2 7 1
3 4 2
4
1 2 1
1 2 1
2 3 2
2 3 2
4
1 2 1
2 3 1
3 4 1
1 4 1
```

Output for Sample Input

```
2
3
5
```

Problem L: Table

Sarina has an **m x n** table whose cells are either 0 or 1. She wants to apply a series of operations on the table in a given order. Each operation involves toggling a subset of cells, where toggling a cell means changing it from 0 to 1, or vice versa. The operations are of the following types:

- *r x*: toggle all cells in every row whose number of 1 cells modulo 2 equals **x**.
- *c x*: toggle all cells in every column whose number of 1 cells modulo 2 equals **x**.

Kindly help Sarina to apply all the given operations and output the final table.

Input

There will be multiple test cases. You must take input until end of file. For each case, the first line of the input contains three positive integers **m**, **n**, and **q**, where **m** and **n** denote the number of rows and columns of the table, respectively **($1 \le m * n \le 10^5$)**, and **q** is the number of operations **($1 \le q \le 10^5$)**. Each of the next **m** lines describes a row of the table as a string of 0s and 1s of length **n**. The next **q** lines describe the list of operations that must be applied to the table in the given order. In all operations, **x** is either 0 or 1.

Output

For each case, print the final table after applying all operations.

Sample Input

```
3 3 2
110
001
111
r 0
c 1
3 3 1
110
001
111
r 0
```

Output for Sample Input

```
110
110
000
001
001
111
```

NB: Large dataset, use faster I/O.

This page is left intentionally blank.

Problem M: Coloring Graph

You are given a connected simple undirected graph with **n** vertices numbered **1** through **n**, and **m** edges. You have to color the graph in two colors numbered **1** and **2**, such that each vertex has exactly one neighbor whose color differs from its own, or claim that there doesn't exist such a coloring. If there are multiple correct answers, you can print any of them.

Input

First line will contain **T (1 ≤ T ≤ 100)**, number of test cases. Then the test cases follow.

The first line of each test case contains two space separated integers **n (1 ≤ n ≤ 1000)** and **m (n-1 ≤ m ≤ n+10)**, the number of vertices and the number of edges in the graph respectively.
- **i**-th of of the next **m** lines contain two integers $\mathbf{a_i}$ and $\mathbf{b_i}$ denoting an edge between vertices $\mathbf{a_i}$ and $\mathbf{b_i}$

You can assume that (i) the sum of **n** over all test cases doesn't exceed **15000** (ii) The graph described is a simple graph, i.e. there are no self loops and multiple edges.(iii) The graph described will be connected, i.e. for any pair of nodes, there exists a path between them.

Output

For each test case:
(i) If no valid coloring exists print **0** on a new line
(ii) If a valid coloring exists, print **1** on a new line and then print **n** space separated integers on another new line, where the **i**-th integer is either **1** or **2** denoting the color of node **i**.

Sample Input

```
2
4 4
1 2
2 3
3 4
4 1
3 3
1 2
2 3
3 1
```

Output for Sample Input

```
1
1 2 2 1
0
```

This page is left intentionally blank.