A. Little Nikita

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

The little boy Nikita was given some cubes as a present. He decided to build a tower out of them.

Initially, the tower doesn't have any cubes. In one move, Nikita either puts exactly 1 cube on top of the tower or removes exactly 1 cube from the top of the tower. Is it possible that after n moves, the resulting tower has exactly m cubes?

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains two integers n and m ($1 \leq n, m \leq 100$).

Output

For each test case, output "Yes" (without quotes) if Nikita can obtain a tower with m cubes, and "No" (without quotes) otherwise.

You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

Standard Input	Standard Output
3 3 3 2 4 5 3	Yes No Yes

Note

In the first test case, Nikita can put 1 cube on top of the tower 3 times in a row, so the answer is "Yes".

In the second test case, Nikita can only end up with either a tower with no blocks or a tower with 2 blocks, so the answer is "No".

,

B. Binary Colouring

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

You are given a positive integer x. Find any array of integers $a_0, a_1, \ldots, a_{n-1}$ for which the following holds:

- $1 \le n \le 32$,
- a_i is 1, 0, or -1 for all $0 \le i \le n-1$,
- $x = \sum_{i=0}^{n-1} a_i \cdot 2^i$,
- There does not exist an index $0 \le i \le n-2$ such that both $a_i \neq 0$ and $a_{i+1} \neq 0$.

It can be proven that under the constraints of the problem, a valid array always exists.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains a single positive integer x ($1 \le x < 2^{30}$).

Output

For each test case, output two lines.

On the first line, output an integer n ($1 \le n \le 32$) — the length of the array $a_0, a_1, \ldots, a_{n-1}$.

On the second line, output the array $a_0, a_1, \ldots, a_{n-1}$.

If there are multiple valid arrays, you can output any of them.

Standard Input	Standard Output
7 1 14 24 15 27 11 19	1 1 5 0 -1 0 0 1 6 0 0 0 -1 0 1 5 -1 0 0 0 1 6 -1 0 -1 0 0 1 5 -1 0 -1 0 1 5 -1 0 1 0 1

Note

In the first test case, one valid array is $[1]$, since $(1) \cdot 2^0 = 1$.

In the second test case, one possible valid array is $[0, -1, 0, 0, 1]$, since $(0) \cdot 2^0 + (-1) \cdot 2^1 + (0) \cdot 2^2 + (0) \cdot 2^3 + (1) \cdot 2^4 = -2 + 16 = 14$.

,

C. Nikita and LCM

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

Nikita is a student passionate about number theory and algorithms. He faces an interesting problem related to an array of numbers.

Suppose Nikita has an array of integers a of length n. He will call a subsequence† of the array *special* if its least common multiple (LCM) is not contained in a. The LCM of an empty subsequence is equal to 0.

Nikita wonders: what is the length of the longest *special* subsequence of a? Help him answer this question!

† A sequence b is a subsequence of a if b can be obtained from a by the deletion of several (possibly, zero or all) elements, without changing the order of the remaining elements. For example, $[5, 2, 3]$ is a subsequence of $[1, 5, 7, 8, 2, 4, 3]$.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 2000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2000$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, output a single integer — the length of the longest *special* subsequence of a.

Standard Input	Standard Output
6 5 1 2 4 8 16 6 3 2 10 20 60 1 7 2 3 4 6 12 100003 1200036 9 2 42 7 3 6 7 7 1 6 8 4 99 57 179 10203 2 11 40812 1 1	0 4 4 5 8 0

Note

In the first test case, the LCM of any non-empty subsequence is contained in a, so the answer is 0.

In the second test case, we can take the subsequence $[3, 2, 10, 1]$, its LCM is equal to 30, which is not contained in a.

In the third test case, we can take the subsequence $[2, 3, 6, 100\,003]$, its LCM is equal to $600\,018$, which is not contained in a.

,

D. XORificator

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

You are given a binary (consisting only of 0s and 1s) $n \times m$ matrix. You are also given a XORificator, using which you can invert all the values in a chosen row (i.e. replace 0 with 1 and 1 with 0).

A column in the matrix is considered *special* if it contains exactly one 1. Your task is to find the maximum number of columns that can be made *special* at the same time, and the set of rows the XORificator should be used on to achieve that.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$, $n \cdot m \le 3 \cdot 10^5$).

Each of the following n lines of the test case contains a binary string of length m.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output two lines.

In the first line, output the maximum number of *special* columns that is possible to get simultaneously.

In the second line, output a binary string of length n, where the i-th character is 0, if you don't use the XORificator on the i-th row, and 1, if you use the XORificator on the i-th row.

If there are multiple valid XORificator configurations that achieve the optimal answer, you can output any of them.

Standard Input	Standard Output
5 3 4 1010 0110 0100 1 1 1 1 1 0 2 5 00101 10110 3 3 101 111	3 010 1 0 1 1 3 00 2 010

000	

Note

In the first test case, you can use the XORificator on the second row to make the columns 2, 3, and 4 *special*.

In the second test case, the only column is already *special*, so you don't need to use the XORificator.

,

E. Tensor

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

This is an interactive problem.

You are given an integer n.

The jury has hidden from you a directed graph with n vertices (numbered from 1 to n) and some number of edges. You additionally know that:

- The graph only contains edges of the form $i \leftarrow j$, where $1 \leq i < j \leq n$.
- For any three vertices $1 \leq i < j < k \leq n$, at least one of the following holds[†]:
  - Vertex i is reachable from vertex j, or
  - Vertex i is reachable from vertex k, or
  - Vertex j is reachable from vertex k.

You want to color each vertex in either black or white such that for any two vertices i and j ($1 \leq i < j \leq n$) of the same color, vertex i is reachable from vertex j.

To do that, you can ask queries of the following type:

- `? i j` — is vertex i reachable from vertex j ($1 \leq i < j \leq n$)?

Find any valid vertex coloring of the hidden graph in at most $2 \cdot n$ queries. It can be proven that such a coloring always exists.

Note that the grader is **not** adaptive: the graph is fixed before any queries are made.

[†] Vertex a is reachable from vertex b if there exists a path from vertex b to vertex a in the graph.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains a single integer n ($3 \leq n \leq 100$) — the number of vertices in the hidden graph.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Interaction

The interaction for each test case begins by reading the integer n.

To make a query, output "`? i j`" without quotes ($1 \leq i < j \leq n$). If vertex i is reachable from vertex j, you will get `YES` as an answer. Otherwise, you will get `NO` as an answer.

If you receive the integer -1 instead of an answer or a valid value of n, it means your program has made an invalid query, has exceeded the limit of queries, or has given an incorrect answer on the previous test case. Your program must terminate immediately to receive a Wrong Answer verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

When you are ready to give the final answer, output "! c_1 c_2 ... c_n" without quotes — the colors of the vertices, where $c_i = 0$ if the vertex is black, and $c_i = 1$ if the vertex is white. After solving all test cases, your program should be terminated immediately.

After printing a query, do not forget to output an end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hacks

To hack, use the following format:

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains two integers n and m ($3 \le n \le 100$, $0 \le m \le \frac{n \cdot (n-1)}{2}$) — the number of vertices and edges in the graph.

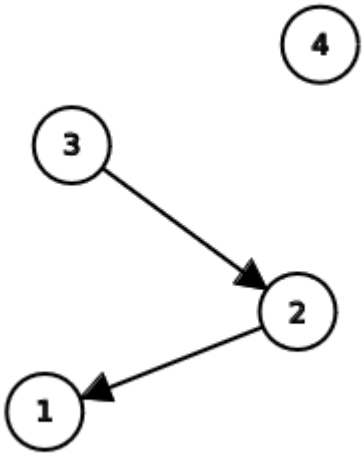
Each of the following m lines should contain two integers a and b ($1 \le b < a \le n$), indicating that there is the edge $a \to b$ in the graph. The graph should satisfy the conditions above.

The sum of n over all test cases should not exceed 1000.

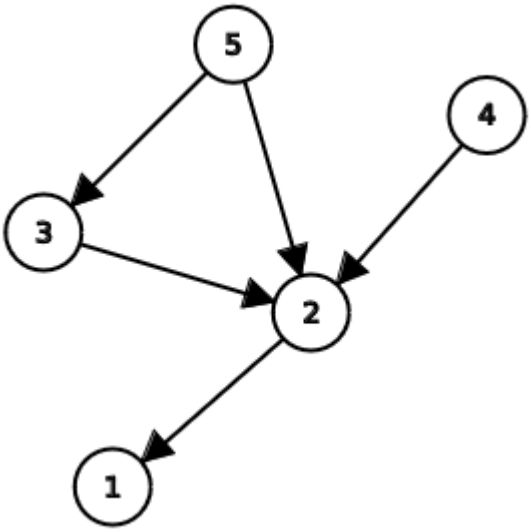
Standard Input	Standard Output
2 4 YES YES YES NO NO NO 5	? 1 2 ? 2 3 ? 1 3 ? 1 4 ? 2 4 ? 3 4 ! 0 0 0 1 ! 1 1 0 1 0

Note

The hidden graph in the first test case:



The hidden graph in the second test case:



The interaction happens as follows:

Solution	Jury	Explanation
	2	There are 2 test cases.
	4	In the first test case, the graph has 4 vertices.
? 1 2	YES	The solution asks if vertex 1 is reachable from vertex 2, and the jury answers YES.
? 2 3	YES	The solution asks if vertex 2 is reachable from vertex 3, and the jury answers YES.
? 1 3	YES	The solution asks if vertex 1 is reachable from vertex 3, and the jury answers YES.
? 1 4	NO	The solution asks if vertex 1 is reachable from vertex 4, and the jury answers NO.
? 2 4	NO	The solution asks if vertex 2 is reachable from vertex 4, and the jury answers NO.
? 3 4	NO	The solution asks if vertex 3 is reachable from vertex 4, and the jury answers NO.
! 0 0 0 1		The solution has somehow determined a valid coloring and outputs it. Since the output is correct, the jury continues to the next test case.

	5	In the second test case, the graph has 5 vertices.
! 1 1 0 1 0		The solution has somehow determined a valid coloring, and outputs it. Since the output is correct and there are no more test cases, the jury and the solution exit.

Note that the line breaks in the example input and output are for the sake of clarity, and do not occur in the real interaction.