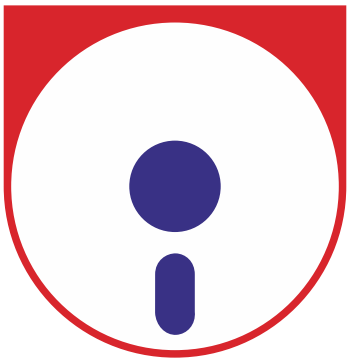


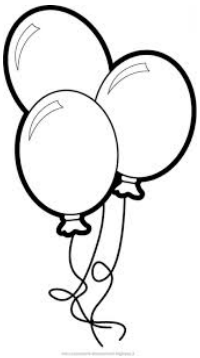
Croatian Open Competition in Informatics

Round 2, November 16th 2019

Tasks

Task	Time limit	Memory limit	Score
ACM	1 second	512 MiB	50
Slagalica	1 second	512 MiB	70
Checker	3 seconds	512 MiB	110
Popcount	1 sekunda	512 MiB	110
Total			340

Task ACM



An ancient programming competition is getting near and it is organized by none other than ACM (*Aeronautic Centre of Metković*). Exactly N teams will compete for the grand prize and among them is a golden Croatian trio: Paula, Marin and Josip. The contest format is standard, while the pilot is performing aerobatic maneuvers, the co-pilot reads the problem statements and attempts to transmit the solutions to the ~~code monkey~~ main programmer who is securely taped to the aircraft's exterior.

The contest consists of M different tasks and the teams are (non-increasingly) ordered by the number of solved tasks.

The teams that have the same number of solved tasks are ordered (non-decreasingly) by so-called penalty time. *Penalty time* of a certain team is the sum of penalty time they achieved on each of their correctly solved tasks. Penalty time of a correctly solved task equals to the time it took for the team to solve that task (from the beginning of the contest) increased by additional **20 minutes** for each of the wrongly submitted solutions for that task. No teams will attempt to submit a solution for a problem that they have already solved and the maximum number of submissions for a certain task is **9 for each team**. If some teams have the same number of solved problems and the same penalty time, they will be ordered alphabetically in the final standings.

The competition lasts for **five hours**. During the first four hours the standings are available to all teams and contain information about the status of each task for each team (number of submissions, whether it was solved and when). During those four hours, the order of the teams will be automatically updated after each submission. In the last hour though, the standings become frozen, i.e., the order of the teams is not updated after a new submission is judged. During that time, each team knows the judgement of their own submissions, but they don't know the judgement of submissions made by other teams. They only know which tasks were submitted by other teams, how many times were they submitted and when was the last submission for each task.

The contest is over and the standings should be unfrozen soon. Our heroes, team named `NijeZivotJedanACM` need your help. They want to know what is the worst possible position on the scoreboard they could end up in after the standings become unfrozen. Help them!

Input

The first line contains integers N ($1 \leq N \leq 1000$) and M ($1 \leq M \leq 15$) from the task description.

The next N lines represent the frozen standings. Each line begins by a team name (string made up of lowercase and uppercase English letters which consists of at most 20 characters, names of all teams will be different) which is separated by a space from M (also space-separated) strings that hold the information about the status of each task for that team.

Those strings are of the form `SX/V`, where:

- `S` represents the status of the task – `'+'` means the task is solved correctly, `'-'` means it is solved incorrectly and `'?'` means that the last submission was sent when the standings were already frozen.
- `X` represents the number of submissions that were sent by that team for this specific task. It is omitted if no submissions were made for that task.
- `V` represents the time when the last submission was sent by that team for this specific task. It is given in the format `HH:MM:SS` (with leading zeroes) and is less than 5 hours. The whole `/V` part is omitted if the task is not solved correctly (status `'-'`)

The last line contains the unfrozen standings of our heroes, team named `NijeZivotJedanACM`.

Output

In the first and only line you should output the worst possible position in the standings where our heroes might end up in after the standings become unfrozen.

Scoring

In test cases worth a total of 20 points, the input won't contain the `'?'` character.

Examples

input

```
2 1
NijeZivotJedanACM -
ZivotJESTJedanACM -
NijeZivotJedanACM -
```

output

```
1
```

input

```
3 2
StoJeZivot ?1/04:00:00 +1/02:04:06
JeLiZivotJedanACM ?1/04:59:59 -
NijeZivotJedanACM ?1/04:42:43 -
NijeZivotJedanACM +1/04:42:43 -
```

output

```
2
```

input

```
7 4
NisamSadaNistaDonio +1/03:59:59 +3/03:42:02 +2/00:14:59 ?1/04:56:12
JeLiMojKockaSeUmio ?4/04:00:00 -3 +1/00:10:01 +9/03:04:42
OstaviDobroJe ?4/04:59:59 -1 +2/00:24:15 +8/03:24:45
DobroJeOstavi +1/01:42:53 - ?9/04:58:23 ?1/04:34:43
NijeZivotJedanACM ?2/04:50:05 ?4/04:32:12 +2/01:32:45 ?1/04:59:59
KoSeToSeta ?1/04:23:32 - +9/01:00:00 -9
SipSipSipSipSipSip - - - ?9/04:00:00
NijeZivotJedanACM -2 +4/04:32:12 +2/01:32:45 +1/04:59:59
```

output

```
3
```

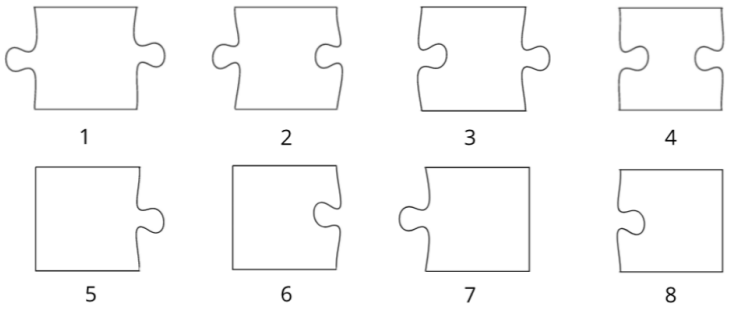
Clarification of the fist example:
Nothing will change after the standings our unfrozen. Therefore, our heroes will remain in the first place!

Clarification of the second example
In the worst case our heroes will only lose from the team `StoJeZivot`, thereby finishing in second place.

Clarification of the third example
In the worst case our heroes will lose from teams U najgorem će slučaju naš trojac `NisamSadaNistaDonio` and `JeLiMojKockaSeUmio`, thereby finishing in third place.

Task Slagalica

Little Fabian got a one-dimensional jigsaw puzzle that consists of N pieces. He quickly realized that each piece belongs to one of the following types:



Additionally, it is known that among those N pieces there is exactly one piece of either type 5 or type 6 (left border) and exactly one piece of either type 7 or type 8 (right border).

Fabian wishes to arrange all of the pieces into a single row such that the first (leftmost) piece is of type 5 or 6 and the last (rightmost) piece is of type 7 or 8. Two pieces can be placed next to each other if and only if their neighbouring borders are of different shapes, i.e., one has a bump (also called *outie* or *tab*) and the other has a hole (also called *innie* or *blank*).

Simply solving the puzzle would be too easy for Fabian so he decided to write a unique positive integer on each of the pieces. Now he is interested in finding the lexicographically smallest solution to the jigsaw puzzle. The solution A is considered lexicographically smaller than solution B if at the first position (from the left) i where they differ it holds that the number written on i-th puzzle in A is smaller than the number written on i-th puzzle in B.

Note: the pieces cannot be rotated.

Input

The first line contains an integer N ($2 \leq N \leq 10^5$) from the task description.

The next N lines contain two integers X_i ($1 \leq X_i \leq 8$) and A_i ($1 \leq A_i \leq 10^9$) which represent the type of the i-th piece and the number Fabian wrote on it. All numbers A_i will be different.

Output

If Fabian cannot solve the jigsaw puzzle, you should output -1 in a single line.

Otherwise, you should output the numbers that are written on the pieces in the lexicographically smallest solution to the puzzle.

Scoring

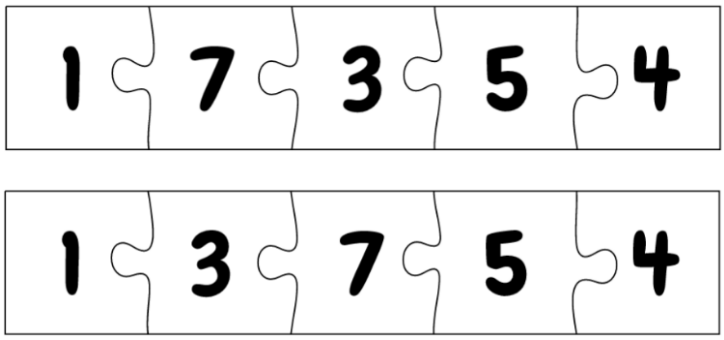
In test cases worth a total of 5 points it will hold $N \leq 4$.
In test cases worth additional 5 points it will hold $N \leq 10$.
In test cases worth additional 10 points pieces of types 2 and 3 will not appear in the input.
In test cases worth additional 20 points there will be at most one piece of type 1 or 4.

If for some test case in which the solution to the puzzle exists, you output the correctly solved puzzle but your solution is not lexicographically smallest, you will get 40% of the points intended for that test case.

Examples

input	input	input
5 1 5 2 7 2 3 8 4 6 1	3 5 1 7 2 4 3	5 2 5 2 7 2 3 8 4 6 1
output	**output**	**output**
1 3 7 5 4	1 3 2	-1

Clarification of the first example:
There are only two possible solutions to the puzzle:



We can see that the second depicted solution has a smaller number written on the second piece. Therefore, that is the lexicographically smallest solution.

Task Checker

"*...fool me once, shame on — shame on you. Fool me — you can't get fooled again.*" – W.

In this task, we will observe regular polygons that have each of their N sides colored in one of three colors and whose vertices are denoted from 1 to N in a clockwise order. A *triangulaton* of a polygon is a decomposition of its area into a set of non-overlapping triangles whose sides either lie on the sides of the polygon or its internal diagonals. Of course, in this task, each of the diagonals used for polygon triangulation is also colored in one of three colors.

The triangulation is said to be *patriotic* if each of its $N-2$ triangles has all three sides of different colors. Your task is to determine whether a given polygon with its diagonals is triangulated and whether that triangulation is patriotic.

Ulazni podaci

The frist line contains the number of subtask this particular test case belongs to (see the table in the scoring section). If your solution doesn't care about that, simply read the number and feel free to ignore it.

The second line contains an integer N from the task description.

The third line contains an integer consisting of N digits which represent the colors of polygon sides. More precisely, the first digit represents the color of side $(1, 2)$, the second digit represents the color of side $(2, 3)$, and so on until the N-th digit which represents the color of side $(N, 1)$. The colors will always be denoted with digits 1, 2 and 3.

Each of the next $N-3$ lines contain a description of one diagonal in the form X Y C, where X and Y are polygon vertices and C is the color of the diagonal ($1 \leq X, Y \leq N, 1 \leq C \leq 3$). Each line describes a valid diagonal, i.e., vertices X and Y will neither be equal nor neighbouring.

Output

If the input polygon is not correctly triangulated, you should output `"neispravna triangulacija"` (invalid triangulation in Croatian).

If the input polygon is correctly triangulated but the triangulation is not patriotic, you should output `"neispravno bojenje"` (invalid coloring in Croatian).

If the input polygon is correctly triangulated and that triangulation is patriotic, you should output `"tocno"` (correct in Croatian).

Scoring

Subtask	Score	Constraints
1	12	$4 \leq N \leq 300$
2	17	$4 \leq N \leq 2000$
3	23	$4 \leq N \leq 2 \cdot 10^5$, the output is either `neispravna triangulacija` or `tocno`
4	23	$4 \leq N \leq 2 \cdot 10^5$, the output is either `neispravno bojenje` or `tocno`
5	35	$4 \leq N \leq 2 \cdot 10^5$

Unlike the task *Trobojnica* from round 1, if your program correctly outputs the first line in each test case of a certain subtask, you will score 100% of the points allocated for that subtask.

Examples

input

```
1
5
12113
1 3 3
2 5 2
```

output

```
neispravna triangulacija
```

input

```
1
4
1212
1 3 2
```

output

```
neispravno bojenje
```

input

```
1
7
1223121
1 3 3
3 5 1
5 7 3
7 3 2
```

output

```
tocno
```

Task Popcount

***Mi**niature **Al**gebraic **Na**tural **R**elay* (also called `MALNAR`) is the latest technological advancement in the flourishing realm of small programmable devices. You can write your own programs for this device using *MalnarScript*, an esoteric programming language with the following set of features:

- Input to the program is a single non-negative integer strictly less than 2^N.
- Output of the program is a single non-negative integer strictly less than 2^N.
- When programming in *MalnarScript* you can only use one N-bit unsigned integer variable A. At the beginning of the program, this variable holds the input and its value at the end of the program is considered to be the program's output.
- The source code of *MalnarScript* must consist of at most K commands of the form `A=<expr>` which are executed in order and each of them must consist of **at most a thousand** characters. The symbol `<expr>` is defined recursively as follows:

```
<expr> = A | <num> | (<expr><operator><expr>)
```

In other words, symbol `<expr>` can either be a variable A, or it can conform to the definition of symbol `<num>`, or it can (inside parentheses) represent a two-membered expression in which each operand conforms to the same `<expr>` definition.

The symbol `<num>` in the definition above represents a non-negative decimal integer strictly less than 2^N, while the symbol `<operator>` can either be `+`, `-`, `|`, `&`, `<<` or `>>` which (in order) represent the operations of addition, subtraction, bitwise or, bitwise and, left shift and right shift.

Also the character `A` can appear `at most 5 times` in the `<expr>` symbol.

In the case of overflow or underflow when performing the operations of addition and subtraction, *MalnarScript* will perform those operations modulo 2^N. For example, when $N = 3$ the expression $(7 + 3)$ will evaluate to 2 and the expression $(2 - 5)$ will evaluate to 5 in `MalnarScript`.

The right side of the equation in each command evaluates into a single number which will then be stored into A. In order to evaluate the right hand side expression, *MalnarScript* first replaces each occurrence of A with its current value. The calculation of expression then proceeds as it would in any mathematical expression, i.e., the parentheses take precedence. Note that the priorities of operators (in term of operation order) are irrelevant because the final result is completely defined by placement of parentheses.

Your task is to write a program which outputs a program in *MalnarScript* which calculates the number of ones in a binary representation of the input value.

Input

The first line contains two integers N and K from the task description.

Output

In the first line you should output the number of commands of the produced *MalnarScript* program.

In the remaining lines you should output the commands of the sought program. Each command must be printed in a separate line and must satisfy the syntax of `MalnarScript` as described in the task description.

It is important that there are no unnecessary empty lines or extra whitespace characters in the output. Each line (including the last) must be terminated by the end-of-line character (`'\n'`).

Scoring

Subtask	Score	Constraints
1	15	$2 \leq N \leq 100$, $K = N - 1$
2	15	$N = 500$, $K = 128$
3	35	$1 \leq N \leq 40$, $K = 7$
4	45	$100 \leq N \leq 500$, $K = 10$

Examples

input

```
2 2
```

output

```
1
A=(A-((A&2)>>1))
```

input

```
3 5
```

output

```
2
A=((A&4)|((A&3)-((A&2)>>1)))
A=((A&3)+((A&4)>>2))
```

Clarification of the first example:
$input = 0 \Rightarrow output = (0 - ((0\&2) >> 1)) = (0 - (0 >> 1)) = (0 - 0) = 0$
$input = 1 \Rightarrow output = (1 - ((1\&2) >> 1)) = (1 - (0 >> 1)) = (1 - 0) = 1$
$input = 2 \Rightarrow output = (2 - ((2\&2) >> 1)) = (2 - (2 >> 1)) = (2 - 1) = 1$
$input = 3 \Rightarrow output = (3 - ((3\&2) >> 1)) = (3 - (2 >> 1)) = (3 - 1) = 2$