Problem A. Acetone, Water and ICPC

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Byteland Hypercomputers, the world's largest computer chip manufacturer, has invented a new class of nanoparticles called Ideal Physical Carbon Particles (ICPCs).

ICPCs are semiconductors. It means that they can be either conductors or insulators of electrons, and thus possess a property that is very important for the computer chip industry.

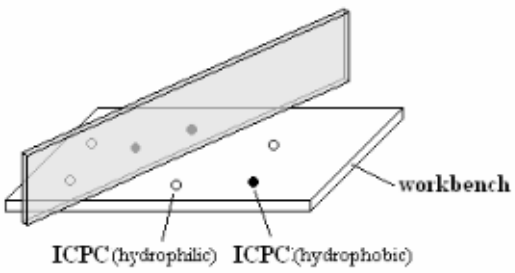
They are also amphiphilic molecules, which means parts of them are hydrophilic while other parts of them are hydrophobic. Hydrophilic ICPCs are soluble in polar solvents (for example, water) but are insoluble in nonpolar solvents (for example, acetone). Hydrophobic ICPCs, on the contrary, are soluble in acetone but insoluble in water. Semiconductor ICPCs dissolved in either water or acetone can be used in the computer chip manufacturing process.

As a materials engineer at Byteland Hypercomputers, your job is to prepare ICPC solutions from ICPC particles. You go to your factory everyday at 8 am and find a batch of ICPC particles on your workbench. You prepare the ICPC solutions by dripping some water, as well as some acetone, into those particles and watch the ICPCs dissolve in the solvents. You always want to prepare unmixed solutions, so you first separate the ICPC particles by placing an Big Analogue Partition Card (BAPC) perpendicular to your workbench. The BAPC is long enough to completely separate the particles. You then drip water on one side of the BAPC and acetone on the other side. The BAPC helps you obtain hydrophilic ICPCs dissolved in water on one side and hydrophobic ICPCs dissolved in acetone on the other side.

If you happen to put the BAPC on top of some ICPC particles, those ICPCs will be right at the border between the water solution and the acetone solution, and they will be dissolved.

Your daily job is very easy and boring, so your supervisor makes it a little bit more challenging by asking you to dissolve as much ICPCs into solution as possible. You know you have to be very careful about where to put the BAPC since hydrophilic ICPCs on the acetone side, or hydrophobic ICPCs on the water side, will not dissolve. As an experienced engineer, you also know that sometimes it can be very difficult to find the best position for the BAPC, so you decide to write a program to help you.

You have asked your supervisor to buy a special digital camera and have it installed above your workbench, so that your program can obtain the exact positions and species (hydrophilic or hydrophobic) of each ICPC particle in a 2D pictures taken by the camera. The BAPC you put on your workbench will appear as a line in the 2D pictures.



Input

There will be no more than 10 test cases. Each case starts with a line containing an integer N, which is the number of ICPC particles in the test case. N lines then follow. Each line contains three integers x, y, r, where (x, y) is the position of the ICPC particle in the 2D picture and r can be 0 or 1, standing for the hydrophilic or hydrophobic type ICPC respectively.

The absolute value of x, y will be no larger than 10 000. You may assume that N is no more than 1000. $N = 0$ signifies the end of the input and need not be processed.

Output

For each test case, output a line containing a single integer, which is the maximum number of dissolved ICPC particles.

Examples

standard input	standard output
3 0 0 0 0 1 0 2 2 1 4 0 0 0 0 4 0 4 0 0 1 2 1 7 -1 0 0 1 2 1 2 3 0 2 1 1 0 3 1 1 4 0 -1 2 0 0	3 3 6

Problem B. Big Boss

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

The *Kiano* company fell into dire financial situation, so the new executive director, as the big boss, decided to completely rebuild the company organization. Until now, it was a typical tree structure:

- There was exactly one executive director, who had no superiors,
- Every other employee had exactly one superior, and there were no cyclic relations.

For every employee x, their *subordinate* is every employee y, who is under x in the tree (there is a sequence of direct superiors from y to x).

After the reform, the company will employ the same people, and it will also be organized as a tree, but every employee will receive a diffrent position, so the shape of the tree may change completely. The executive director is, however, guaranteed to keep the position. Now everyone is afraid to give orders to anyone else – any moment now, a subordinate may become the superior...

Given the description of the tree before and after the takover, for every employee x determine the number of the people who were subordinates of x and will remain the subordinates after the takeover.

Input

The first line of the input contains an integer n ($2 \le n \le 200\,000$): the number of employees. The employees are numbered from 1 to n, with the person number 1 being the executive director. The second line contains the company structure before the reform: there are $n-1$ numbers $a_2, a_3, \ldots, a_n$ with a_i being the number of i's superior. The third line also contains $n-1$ numbers $b_2, b_3, \ldots, b_n$, where b_i is the superior of i after the takeover. You may assume that both descriptions define a proper tree rooted at 1.

Output

Output a single line containing n numbers – i-th of them should be the number of people who are i's subordinates in both trees at once.

Example

standard input	standard output
5 1 1 3 3 1 2 3 1	4 0 1 0 0

Problem C. Circular Task

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There are N circular stones on the plane and two circular openings A and B.

Your task is to check if it is possible to choose one point in A and another point in B so that the segment connecting these points does not intersect any of the stones.

Input

The first line of the input contains the number of stones N ($0 \le N \le 256$). Each of the following N lines contains three integers — the coordinates of the center of the corresponding stone and its radius, respectively. The last two lines contain the coordinates of the center and radius of the two openings.

All coordinates do not exceed 10^4 by absolute value. Stones and openings cannot overlap but may touch each other.

It is guaranteed that if solution exists then it is possible to choose points inside the corresponding openings so that the segment connecting these points is no closer than 10^{-6} meters to any stone.

Output

Output "YES" (without quotes) if it is possible to choose appropriate positions in the openings (as described above). Otherwise, output "NO" (without quotes).

Examples

standard input	standard output
1 0 0 2 0 4 1 0 -4 1	NO
1 0 0 1 0 4 2 0 -4 2	YES

Problem D. Diary

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

In Byteland standard notions of day, month, and year are not used to designate dates. Instead, each date is represented by a single non-negative integer (denoting the number of days passed from the begining of New Byteland Era). One of the citizens, Byteazar, has been recording important historial events that took place during his long life. In this diary each event is marked with a date at which this particular event has occured. Now Byteasar has an idea to publish these chronicles. The diary consist of N records corresponding to dates $a_1 < a_2 < \ldots < a_N$.

Since the number of records might be huge the publisher insists the dates to be pre-processed and presented in a "compact" form. Namely, the following properties must be fulfilled:

1. The first and the last dates (a_1 and a_N) must remain unchanged.
2. Each other date a_i ($1 < i < N$) may be modified: some (possibly none) leading digits may be dropped or some number of leading zeroes (possibly none) may be added. It is not allowed to remove leading digits from a date and simultaneously add new leading zeroes to it. It is also not allowed to remove all digits from a date.
3. The initial increasing sequence α should be uniquely determined by a resulting one β. That is, it should not be possible to obtain β (under the rules explained above) from any other initial sequence $\alpha' \neq \alpha$.
4. The total number of digits in all dates of the "compact" form should be as small as possible.

By these rules the total number of digits in a "compact" form may be greater than number of digits in the initial sequence.

Help Byteasar: write a program that calculates the total size a "compact" form of the initial sequence $a_1, \ldots, a_N$.

Input

The first line of the input conatains a single integer N ($2 \le N \le 1\,000$). Next, N lines follow, describing an increasing sequence $a_1, \ldots, a_N$ ($0 < a_i < 2 \cdot 10^9$ for each i). None of a_i contains leading zeroes.

Output

Print out a single integer indicating the difference between the total number of digits in a "compact" form of the sequence and that number for the initial sequence.

Example

standard input	standard output
3 10 11 12	1
4 107 213 789 999	0
5 109 117 124 133 140	6

Note

A "compact" form for the first example is:

```
10
 1
12
```

In the second example a "compact" form coincides with the initial sequence.

In the third example a "compact" form is:

```
109
  7
  4
  3
140
```

Problem E. Example

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

As you may have probably noticed a problem statement in a programming contest consists of several sections. The most important section is, of course, the "Example" section. Some seasoned contestants even start reading the problem statement from the examples. And, unfortunately, the least read section is the problem description section. It is quite disappointing for the problem authors because they feel that their writing skills are largely wasted.

So the authors decided to describe examples in the same language, as the rest of the problem statement using the following rules.

- Natural integer numbers shall be written in plain English. The numbers shall be less than one hundred. The designator `number` shall be written before numbers, except when the corresponding number is used as a *repetition factor*. For example, `number zero`, `number sixteen`, or `number sixty one`.
- Sequences of characters (strings) shall be written either in quotes, or in apostrophes, for example `"John's pen"`, or `'5" tall'`. Note, that `'` may be used in strings enclosed in `"` and vice versa. The designator `string` shall be written before strings, for example `string 'Hello'`.
- The designator `space` denotes one space character.
- The English numerals from 1 to 19 and from 20 to 90 are spelled as follows: "*one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen twenty thirty forty fifty sixty seventy eighty ninety*"
- A number, string or space may be prefixed with a *repetition factor*. The repetition factor is a number greater than one. The designator after the repetition factor is written in plural form. For example, `four numbers five`, or `six strings 'A'`. If the repetition factor is used for numbers, the numbers are separated with one space character. So, the former example means `5 5 5 5`, but the latter example — `AAAAAA`.
- Let the numbers, strings and spaces with possible repetition factors be called *fragments*. Fragments may be organized into *sequences* using the `followed by` copulative. For example, `number five followed by number six`. One implicit space character is assumed between numbers, a number and a string, and a string and a number so the example above means `5 6`.
- An example is described line by line. The first line is always described as `The first line...` The following lines are described either as `The next line...` or as `The next # lines...`, where `#` is a number greater than one. Empty lines are described as `is empty` or `are empty`. Other lines are described as `contains` or `contain` followed by sequences. The first letter of a sentence is capitalized. The sentence is terminated by a full stop (`.`). The full stop is not separated by space from the preceding word, but is separated by at least one space from the next word.

Input

The input contains a free-flow text describing an example. Words are separated by an arbitrary number of spaces and newlines. There are no whitespace characters after the last full stop. The total size of the input will be no greater than 1 MiB.

Output

The output should contain the decoded example. The total size of the output shall be no greater than 1 MiB.

Example

standard input	standard output
The first line contains four numbers twenty eight. The next line is empty. The next two lines contain six strings '-'. The next line contains number four followed by number seventy seven followed by string 'meat' followed by three strings "!".	28 28 28 28 ------ ------ 4 77 meat!!!

Problem F. Find The Integer

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

Your task is to guess some secret positive integer N.

Initially, the value of the variable v in the memory of the jury program is N.

Each time jury program tells you the sum of digits if decimal representation of the current value of v. For one query you may ask the jury program to decrease v by k, where k is positive integer. If the difference become negative, you fail immediately. Otherwise jury program decreases v by k and tells you sum of digits of the new value of v.

When your program have enough information to guess N, it shall tell the value of N to the jury program. If your guess is right, you never failed and used no more than 128 queries (note that naming the answer is not counted for query), then your solution is passed the test case.

Interaction Protocol

At the beginning of the interaction the jury program prints one integer — sum of digits in the decimal representation of the integer N (and sets $v = N$).

Then you may do the queries. Each query have the form "`?` k", where k is positive integer, does not exceeding 10^{18}. It means that v shall be decreased by k.

If the query have wrong format, or $v - k$ become negative, or there are more than 128 queries used, interaction immediately stops and the test considered failed. Otherwise value of v is decreased by k, and jury program tells you sum of digits in the decimal representation of the new value of v.

If you want to print the answer, print "`!` X", where X is guessed value of N. You may assume that N does not exceeds 10^{18} and that jury program is not adaptive (i.e. never changes N in the process of guessing). This action does not counted as query.

Dont forget to print after each of your actions exactly one newline character and flush the output buffer using the "`flush`" function of your programming language. Otherwise your solution will receive "Idleness Limit Exceeded" verdict.

Example

standard input	standard output
3 2 0	 ? 1 ? 11 ! 12

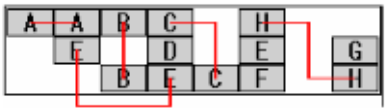
Note

Note that in case $N = 3$ query "? 11" cause immediate fail.

Problem G. Game with Blocks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Recently, George found a new game on the Internet. The game rule is very simple. Initially, a board of size $n \times m$ is filled with $n \times m$ blocks. Each of these blocks has a symbol on it. All you need to do is to find a pair of blocks with the same symbol on them, which can be linked with a line that consists of at most three straight horizontal or vertical line segments. Note that the line segments cannot cross the other blocks on the board (see figure below for some examples of possible links, note that some blocks have been already removed from the board).



If you successfully find such a pair of blocks, the two blocks can be popped (that is, removed) together. After this, some of the blocks may be moved to new positions on the board following the rules described later.

Then, you can start to find the next pair. The game continues until there are no block left on the board or you cannot find such a pair. The blocks are moved according to the following rules. First, each block have a static moving attribute, which is one of 'up', 'down', 'left', 'right' and 'stand still'.

After a pair of block is removed, the blocks are checked one by one to see whether they can be moved towards the direction of its moving attribute. The blocks in the top row are checked first. Inside the same row, the blocks on the left are checked first. If the adjacent position at the direction of the block's moving attribute is not occupied, the block will be moved to that position immediately. No block can be moved beyond the boundary of the game board. Of course, a block with attribute 'stand still' will always stay at its original position. After all the blocks are checked, which is called a turn of checking, another turn of checking is started. This continues until no more blocks can be moved to a new position following the moving rules. Note that inside each turn of checking, each of the blocks is checked and possibly moved only once. Blocks must not be checked and moved on its new position in one turn of checking.

George felt that the game was very interesting. However, after some time of playing, he found that when the size of the board is rather large, finding a pair of block becomes a very tough work. Further more, he often gets a 'Game Over' because of no more blocks can be popped. Robert felt that it is not his fault that not all the blocks are being popped. It is only that there is a great chance that the game cannot be finished if the blocks are placed randomly at first. However, it will be very time consuming to prove this by playing the game many times. So, George asks you to write a program for him that will simulate his behavior in the game and see if the game can be finished. In order to make such a program possible, George summarizes his rules of selecting block pairs as follows. First, the pair of blocks that can be linked with one straight line segment must be found and popped first, because such kind of pairs are easy to find. Next, if such a pair does not exist, the pairs that can be linked by two straight line segments must be found and popped. Finally, if both of the two kinds of pairs do not exist, the pairs that can be linked by three straight line segments must be found and popped. If more than one pair that can be linked with the same number of straight line segments exists, the pair that contains a block, which is positioned at the most top row (or most left if two more blocks are positioned in the same row), will be selected first. If this rule still cannot break the tie (more than one pair may share one block that is positioned at the most top, left position), the other block in these pairs are compared according to the same rules. Figure below shows a trace of a game that follows the above rules.

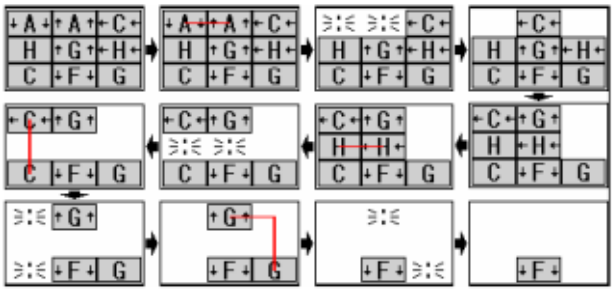


Fig 2.

Input

The input contains 32 or less test cases. The first line of each test case contains 2 integers n and m ($1 \le n, m \le 30$), which is the size of the board.

After this line, there will be n more lines. Each of these lines contains m strings, separated by single spaces. Each of these strings represents one block in the initial configuration. Each string always consists of two capital letters. The first letter is the symbol of the block. The second letter is always one of the letters 'U','D','L', 'R' and 'S', which shows the block's moving attribute: up, down, left, right, and stand still respectively. There are no blank lines between test cases. The input ends with a line of two 0s: 0 0.

Output

For each test case, first output the test case number. After this line, you must output the final configuration of the board with n lines, each containing m characters. If there is a block on the position, output the symbol of the block. If there is no block on the position, output a period instead. Do not output blank lines between test cases.

Examples

standard input	standard output
3 3 AD AU CL HS GU HL CS FD GS 1 2 BS BL 0 0	Case 1 ... ... .F. Case 2 ..

Problem H. Hieroglyphs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Archaeologists have just deciphered hieroglyphs on walls of a pyramid. The writings on one of the walls describe N sacred numbers. All numbers which are divisible by at least one of these numbers are also sacred.

The writings on M other walls claim that the Q_i-th lowest sacred number has magic properties. The archaeologists would like to know which numbers have the magic properties. Could you help them with that?

You are given N positive integers $A_1, A_2, \ldots, A_N$ and M positive integers $Q_1, Q_2, \ldots, Q_M$. For each $i \in \{1, 2, \ldots, M\}$ find the Q_i-th lowest positive integer which is divisible by at least one of the integers $A_1, A_2, \ldots, A_N$.

Input

The first line of the input contains two integers N a M. The second line contains space-separated integers A_1, A_2, ..., A_N. Then, M lines follow. Each of them contains an integer Q_i.

It holds $1 \le N \le 15$ and $1 \le M \le 50$.
For all $i \in \{1, 2, \ldots, N\}$ it holds $2 \le A_i \le 10^{18}$.
For the product of these numbers it holds $A_1 \cdot A_2 \cdot \ldots \cdot A_N \le 10^{18}$.
For all $i \in \{1, 2, \ldots, M\}$ it holds $1 \le Q_i \le 10^{18}$.
Each number on the output is lower than or equal to 10^{18}.

Output

Output M lines. The i-th line should contain the Q_i-th lowest positive integer which is divisible by at least one of the integers $A_1, A_2, \ldots, A_N$.

Example

standard input	standard output
5 5 2 5 7 10 11 1 2 3 10 20	2 4 5 14 28
2 1 70 100 5	210

Problem I. Incident in Bytesburg

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 seconds
Memory limit:	512 mebibytes

There was a criminal incident in a city of Bytesburg. The thief successfully escaped and hid somewhere in the city. Byteazar is investigating this crime. His goal is to find and arrest the thief.

The city consists of N houses and $N-1$ roads, connecting some of the houses in such a way, that there exists a unique path between any two houses (i.e. the city forms a tree structure). The thief is hiding in one of the houses.

To locate the thief, Byteazar can choose a house h and search it. If it was the house where the thief was hiding, Byteazar arrests him. Otherwise he interrogates the inhabitants of the house and they provide him with the following information: "If you picture the city as a rooted tree, with house h as its root and houses $c_1, c_2 \dots c_m$ as the children of h, then the thief is hiding in one of the houses of the subtree rooted at c_i (for some i, $1 \le i \le m$)."

Byteazar have to keep searching the houses until you find and arrest the thief. You can suppose that the thief stays hidden in the same house during the whole investigation process (i.e. he doesn't change the location).

Obviously, the order in which Byteasar searches the houses matters, because even if he don't find the thief in a house, with the provided piece of information he can highly reduce the number of possible houses where the thief can be hiding.

You are given the description of the city. Come up with a strategy for searching the houses, that minimizes the number of houses Byteasar need to search in the worst possible scenario.

Input

First line contains a single integer N ($2 \le N \le 10^5$), the number of houses in the city (houses are numbered from 0 to $N-1$).

Second line contains $N-1$ space separated integers, $v_1 v_2 \dots v_{N-1}$. Integer v_i ($1 \le i \le N-1$) means there is a road connecting the houses with numbers v_i and i ($v_i < i$).

Output

Output exactly one integer, the number of houses you need to search in the worst possible scenario, when searching using the optimal strategy.

Examples

standard input	standard output
5 0 1 1 1	2
8 0 1 2 1 3 5 6	3

Problem J. James Bond

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

James Bond just got a new mission. His task is to take a picture of a top-secret object. It is a building having the form of a regular polygon with N vertices. Each side has the length of L meters.

The agent has to take a picture with maximum possible detail level; to this aim he should be standing as close to the building as possible (the distance to building is measured as the minimum length of a segment connecting the agent's position with a point from building).

It's additionally required that the whole building must fit into a picture. It doesn't matter which part of the building will be pictured. Bond has a secret button-mounted camera with horizontal angle of vision equal to A degrees. Vertical angle of vision and the height of the building require the picture to be taken from the distance of at least H meters. Bond can choose an arbitrary position for his shot and rotate his camera in horizontal plane but is not allowed to perform any rotations in vertical plane.

You are asked to write a program to help him by choosing the closest possible location for the shot. For security purposes you should print the minimum distance to the building, not the location itself.

Input

The input consists of a single line with integers N, L, H, A ($3 \le N \le 17$, $3 \le L, H \le 1000$, $30 \le A \le 180$).

Output

Print out the required minimum distance to the building, in meters. An absolute error of up to 10^{-3} is allowed.

Examples

standard input	standard output
4 100 30 90	30.000
3 10 5 30	10.000

Problem K. King and Highways

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The Byteland Kingdom consists of N cities connected by M highways. Each highway is bidirectional and connects a pair of distinct cities. Some pairs of distinct cities may be connected by more than one highway.

As the maintenance cost of these highways became infeasible high, the King has decided some of the highways to go private. A number of private companies agreed to participate in the tender. These companies have reached the mutual agreement, so that for every highway it is known by now which company is willing to obtain it.

The King insists that the following additional requirements must be fulfilled:

1. To encourage competition and avoid monopolization, no company should be allowed to privatize more than one highway.
2. Since most of citizens do not like to think much as they drive, it is required that for any two distinct cities there should be at most one route connecting them and passing exclusively through the privatized highways.

Your task is to find which highways should go private (satisfying the above conditions) in order to maximize the total number of privatized highways.

Input

The first line of the input contains integers N and M ($1 \le N \le 100$, $0 \le M \le 5\,000$). The following M lines contain the descriptions of highways, one per line. The i-th line ($1 \le i \le M$) has the form $a_i\ b_i\ c_i$, where a_i and b_i are the cities which are connected by the highway i and c_i is the company which may privatize this highway ($1 \le a_i \ne b_i \le N$, $1 \le c_i \le 100$).

Output

On the first line output the maximum number K of highway which can be made private. On the second line output K integers — the numbers of these highways. Each privatized highway should be listed exactly once.

If there are several possible solutions you may output any of them.

Example

standard input	standard output
4 5 1 2 1 3 1 1 2 3 1 1 4 2 3 4 3	3 1 4 5
4 6 1 2 1 2 1 2 2 3 1 3 2 2 3 4 1 4 3 2	2 4 1