Problem A
Ginkgo Numbers

Input: Standard Input
Time Limit: 30 seconds

We will define Ginkgo numbers and multiplication on Ginkgo numbers.

A *Ginkgo number* is a pair $\langle m, n \rangle$ where m and n are integers. For example, $\langle 1, 1 \rangle$, $\langle -2, 1 \rangle$ and $\langle -3, -1 \rangle$ are Ginkgo numbers.

The multiplication on Ginkgo numbers is defined by $\langle m, n \rangle \cdot \langle x, y \rangle = \langle mx - ny,\ my + nx \rangle$. For example, $\langle 1, 1 \rangle \cdot \langle -2, 1 \rangle = \langle -3, -1 \rangle$.

A Ginkgo number $\langle m, n \rangle$ is called a *divisor* of a Ginkgo number $\langle p, q \rangle$ if there exists a Ginkgo number $\langle x, y \rangle$ such that $\langle m, n \rangle \cdot \langle x, y \rangle = \langle p, q \rangle$.

For any Ginkgo number $\langle m, n \rangle$, Ginkgo numbers $\langle 1, 0 \rangle$, $\langle 0, 1 \rangle$, $\langle -1, 0 \rangle$, $\langle 0, -1 \rangle$, $\langle m, n \rangle$, $\langle -n, m \rangle$, $\langle -m, -n \rangle$ and $\langle n, -m \rangle$ are divisors of $\langle m, n \rangle$. If $m^2 + n^2 > 1$, these Ginkgo numbers are distinct. In other words, any Ginkgo number such that $m^2 + n^2 > 1$ has at least eight divisors.

A Ginkgo number $\langle m, n \rangle$ is called a *prime* if $m^2 + n^2 > 1$ and it has exactly eight divisors. Your mission is to check whether a given Ginkgo number is a prime or not.

The following two facts might be useful to check whether a Ginkgo number is a divisor of another Ginkgo number.

- Suppose $m^2 + n^2 > 0$. Then, $\langle m, n \rangle$ is a divisor of $\langle p, q \rangle$ if and only if the integer $m^2 + n^2$ is a common divisor of $mp + nq$ and $mq - np$.
- If $\langle m, n \rangle \cdot \langle x, y \rangle = \langle p, q \rangle$, then $(m^2 + n^2)(x^2 + y^2) = p^2 + q^2$.

Input

The first line of the input contains a single integer, which is the number of datasets.

The rest of the input is a sequence of datasets. Each dataset is a line containing two integers m and n, separated by a space. They designate the Ginkgo number $\langle m, n \rangle$. You can assume $1 < m^2 + n^2 < 20000$.

Output

For each dataset, output a character '`P`' in a line if the Ginkgo number is a prime. Output a character '`C`' in a line otherwise.

Sample Input

```
8
10 0
0 2
-3 0
4 2
0 -13
-4 1
-2 -1
3 -1
```

Output for the Sample Input

```
C
C
P
C
C
P
P
C
```

Problem B
Stylish

Input: Standard Input
Time Limit: 30 seconds

Stylish is a programming language whose syntax comprises *names*, that are sequences of Latin alphabet letters, three types of *grouping symbols*, periods (‘.’), and newlines. Grouping symbols, namely round brackets (‘(’ and ‘)’), curly brackets (‘{’ and ‘}’), and square brackets (‘[’ and ‘]’), must match and be nested properly. Unlike most other programming languages, Stylish uses periods instead of whitespaces for the purpose of term separation. The following is an example of a Stylish program.

```
1 (Welcome.to
2 .........Stylish)
3 {Stylish.is
4 .....[.(a.programming.language.fun.to.learn)
5 .......]
6 .....Maybe.[
7 .......It.will.be.an.official.ICPC.language
8 .......]
9 .....}
```

As you see in the example, a Stylish program is indented by periods. The *amount of indentation* of a line is the number of leading periods of it.

Your mission is to visit Stylish masters, learn their indentation styles, and become the youngest Stylish master. An indentation style for *well-indented* Stylish programs is defined by a triple of integers, (R, C, S), satisfying $1 \leq R, C, S \leq 20$. R, C and S are amounts of indentation introduced by an open round bracket, an open curly bracket, and an open square bracket, respectively.

In a well-indented program, the amount of indentation of a line is given by $R(r_o - r_c) + C(c_o - c_c) + S(s_o - s_c)$, where r_o, c_o, and s_o are the numbers of occurrences of open round, curly, and square brackets in all preceding lines, respectively, and r_c, c_c, and s_c are those of close brackets. The first line has no indentation in any well-indented program.

The above example is formatted in the indentation style $(R, C, S) = (9, 5, 2)$. The only grouping symbol occurring in the first line of the above program is an open round bracket. Therefore the amount of indentation for the second line is $9 \cdot (1 - 0) + 5 \cdot (0 - 0) + 2 \cdot (0 - 0) = 9$. The first four lines contain two open round brackets, one open curly bracket, one open square bracket, two

close round brackets, but no close curly nor square bracket. Therefore the amount of indentation for the fifth line is $9 \cdot (2-2) + 5 \cdot (1-0) + 2 \cdot (1-0) = 7$.

Stylish masters write only well-indented Stylish programs. Every master has his/her own indentation style.

Write a program that imitates indentation styles of Stylish masters.

Input

The input consists of multiple datasets. The first line of a dataset contains two integers $p\,(1 \leq p \leq 10)$ and $q\,(1 \leq q \leq 10)$. The next p lines form a well-indented program P written by a Stylish master and the following q lines form another program Q. You may assume that every line of both programs has at least one character and at most 80 characters. Also, you may assume that no line of Q starts with a period.

The last dataset is followed by a line containing two zeros.

Output

Apply the indentation style of P to Q and output the appropriate amount of indentation for each line of Q. The amounts must be output in a line in the order of corresponding lines of Q and they must be separated by a single space. The last one should not be followed by trailing spaces. If the appropriate amount of indentation of a line of Q cannot be determined uniquely through analysis of P, then output `-1` for that line.

Sample Input

```
5 4
(Follow.my.style
.........starting.from.round.brackets)
{then.curly.brackets
.....[.and.finally
.......square.brackets.]}
(Thank.you
{for.showing.me
[all
the.secrets]})
4 2
(This.time.I.will.show.you
.........(how.to.use.round.brackets)
.........[but.not.about.square.brackets]
.........{nor.curly.brackets})
(I.learned
how.to.use.round.brackets)
4 2
```

```
(This.time.I.will.show.you
.........(how.to.use.round.brackets)
.........[but.not.about.square.brackets]
.........{nor.curly.brackets})
[I.have.not.learned
how.to.use.square.brackets]
2 2
(Be.smart.and.let.fear.of
..(closed.brackets).go)
(A.pair.of.round.brackets.enclosing
[A.line.enclosed.in.square.brackets])
1 2
Telling.you.nothing.but.you.can.make.it
[One.liner.(is).(never.indented)]
[One.liner.(is).(never.indented)]
2 4
([{Learn.from.my.KungFu
...}])
((
{{
[[
]]}}))
1 2
Do.not.waste.your.time.trying.to.read.from.emptiness
(
)
2 3
({Quite.interesting.art.of.ambiguity
....})
{
(
)}
2 4
({[
.....................................................]})
(
{
[
]})
0 0
```

Output for the Sample Input

```
0 9 14 16
0 9
0 -1
```

```
0 2
0 0
0 2 4 6
0 -1
0 -1 4
0 20 40 60
```

Problem C
One-Dimensional Cellular Automaton

Input: Standard Input
Time Limit: 30 seconds

There is a one-dimensional cellular automaton consisting of N cells. Cells are numbered from 0 to $N-1$.

Each cell has a state represented as a non-negative integer less than M. The states of cells evolve through discrete time steps. We denote the state of the i-th cell at time t as $S(i,t)$. The state at time $t+1$ is defined by the equation

$$S(i,t+1) = \Big(A \times S(i-1,t) + B \times S(i,t) + C \times S(i+1,t)\Big) \bmod M, \tag{1}$$

where A, B and C are non-negative integer constants. For $i < 0$ or $N \leq i$, we define $S(i,t) = 0$.

Given an automaton definition and initial states of cells, your mission is to write a program that computes the states of the cells at a specified time T.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

$$\begin{array}{l} N \;\; M \;\; A \;\; B \;\; C \;\; T \\ S(0,0) \;\; S(1,0) \;\; \ldots \;\; S(N-1,0) \end{array}$$

The first line of a dataset consists of six integers, namely N, M, A, B, C and T. N is the number of cells. M is the modulus in the equation (1). A, B and C are coefficients in the equation (1). Finally, T is the time for which you should compute the states.

You may assume that $0 < N \leq 50$, $0 < M \leq 1000$, $0 \leq A, B, C < M$ and $0 \leq T \leq 10^9$.

The second line consists of N integers, each of which is non-negative and less than M. They represent the states of the cells at time zero.

A line containing six zeros indicates the end of the input.

Output

For each dataset, output a line that contains the states of the cells at time T. The format of the output is as follows.

$$S(0,T) \;\; S(1,T) \;\; \ldots \;\; S(N-1,T)$$

Each state must be represented as an integer and the integers must be separated by a space.

Sample Input

```
5 4 1 3 2 0
0 1 2 0 1
5 7 1 3 2 1
0 1 2 0 1
5 13 1 3 2 11
0 1 2 0 1
5 5 2 0 1 100
0 1 2 0 1
6 6 0 2 3 1000
0 1 2 0 1 4
20 1000 0 2 3 1000000000
0 1 2 0 1 0 1 2 0 1 0 1 2 0 1 0 1 2 0 1
30 2 1 0 1 1000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
30 2 1 1 1 1000000000
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
30 5 2 3 1 1000000000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0
```

Output for the Sample Input

```
0 1 2 0 1
2 0 0 4 3
2 12 10 9 11
3 0 4 2 1
0 4 2 0 4 4
0 376 752 0 376 0 376 752 0 376 0 376 752 0 376 0 376 752 0 376
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 3 2 2 2 3 3 1 4 3 1 2 3 0 4 3 3 0 4 2 2 2 2 1 1 2 1 3 0
```

Problem D
Find the Outlier

Input: Standard Input
Time Limit: 30 seconds

Professor Abacus has just built a new computing engine for making numerical tables. It was designed to calculate the values of a polynomial function in one variable at several points at a time. With the polynomial function $f(x) = x^2 + 2x + 1$, for instance, a possible expected calculation result is 1 (= $f(0)$), 4 (= $f(1)$), 9 (= $f(2)$), 16 (= $f(3)$), and 25 (= $f(4)$).

It is a pity, however, the engine seemingly has faulty components and exactly one value among those calculated simultaneously is always wrong. With the same polynomial function as above, it can, for instance, output 1, 4, 12, 16, and 25 instead of 1, 4, 9, 16, and 25.

You are requested to help the professor identify the faulty components. As the first step, you should write a program that scans calculation results of the engine and finds the wrong values.

Input

The input is a sequence of datasets, each representing a calculation result in the following format.

$$
\begin{array}{l}
d \\
v_0 \\
v_1 \\
\vdots \\
v_{d+2}
\end{array}
$$

Here, d in the first line is a positive integer that represents the degree of the polynomial, namely, the highest exponent of the variable. For instance, the degree of $4x^5 + 3x + 0.5$ is five and that of $2.4x + 3.8$ is one. d is at most five.

The following $d + 3$ lines contain the calculation result of $f(0)$, $f(1)$, ..., and $f(d + 2)$ in this order, where f is the polynomial function. Each of the lines contains a decimal fraction between -100.0 and 100.0, exclusive.

You can assume that the wrong value, which is exactly one of $f(0)$, $f(1)$, ..., and $f(d+2)$, has an error greater than 1.0. Since rounding errors are inevitable, the other values may also have errors but they are small and never exceed 10^{-6}.

The end of the input is indicated by a line containing a zero.

Output

For each dataset, output i in a line when v_i is wrong.

Sample Input

```
2
1.0
4.0
12.0
16.0
25.0
1
-30.5893962764
5.76397083962
39.3853798058
74.3727663177
4
42.4715310246
79.5420238202
28.0282396675
-30.3627807522
-49.8363481393
-25.5101480106
7.58575761381
5
-21.9161699038
-48.469304271
-24.3188578417
-2.35085940324
-9.70239202086
-47.2709510623
-93.5066246072
-82.5073836498
0
```

Output for the Sample Input

```
2
1
1
6
```

Problem E
Sliding Block Puzzle

Input: Standard Input
Time Limit: 30 seconds

In sliding block puzzles, we repeatedly slide pieces (blocks) to open spaces within a frame to establish a goal placement of pieces.

A puzzle creator has designed a new puzzle by combining the ideas of sliding block puzzles and mazes. The puzzle is played in a rectangular frame segmented into unit squares. Some squares are pre-occupied by obstacles. There are a number of pieces placed in the frame, one 2×2 king piece and some number of 1×1 pawn pieces. Exactly two 1×1 squares are left open. If a pawn piece is adjacent to an open square, we can slide the piece there. If a whole edge of the king piece is adjacent to two open squares, we can slide the king piece. We cannot move the obstacles. Starting from a given initial placement, the objective of the puzzle is to move the king piece to the upper-left corner of the frame.

The following figure illustrates the initial placement of the fourth dataset of the sample input.

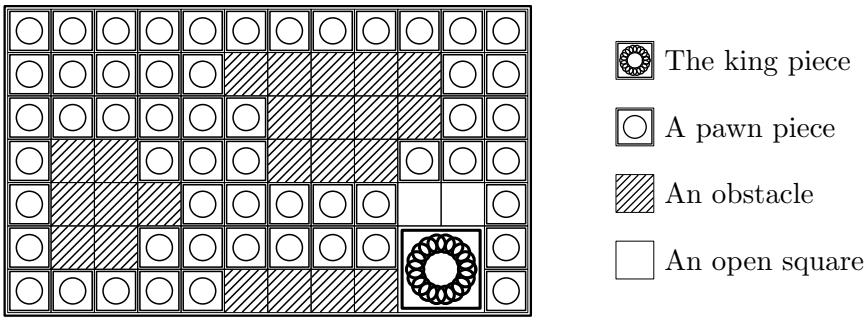

Figure E.1: The fourth dataset of the sample input.

Your task is to write a program that computes the minimum number of moves to solve the puzzle from a given placement of pieces. Here, one move means sliding either king or pawn piece to an adjacent position.

Input

The input is a sequence of datasets. The first line of a dataset consists of two integers H and W separated by a space, where H and W are the height and the width of the frame. The following H lines, each consisting of W characters, denote the initial placement of pieces. In those H lines, '`X`', '`o`', '`*`', and '`.`' denote a part of the king piece, a pawn piece, an obstacle, and an open square, respectively. There are no other characters in those H lines. You may assume that $3 \leq H \leq 50$ and $3 \leq W \leq 50$.

A line containing two zeros separated by a space indicates the end of the input.

Output

For each dataset, output a line containing the minimum number of moves required to move the king piece to the upper-left corner. If there is no way to do so, output `-1`.

Sample Input

```
3 3
oo.
oXX
.XX
3 3
XXo
XX.
o.o
3 5
.o*XX
oooXX
oooo.
7 12
oooooooooooo
ooooo*****oo
oooooo****oo
o**ooo***ooo
o***ooooo..o
o**ooooooXXo
ooooo****XXo
5 30
oooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooo
o**************************oo
XX.ooooooooooooooooooooooooooo
XX.ooooooooooooooooooooooooooo
0 0
```

Output for the Sample Input

```
11
0
-1
382
6807
```

Problem F
Never Wait for Weights

Input: Standard Input
Time Limit: 15 seconds

In a laboratory, an assistant, Nathan Wada, is measuring weight differences between sample pieces pair by pair. He is using a balance because it can more precisely measure the weight difference between two samples than a spring scale when the samples have nearly the same weight.

He is occasionally asked the weight differences between pairs of samples. He can or cannot answer based on measurement results already obtained.

Since he is accumulating a massive amount of measurement data, it is now not easy for him to promptly tell the weight differences. Nathan asks you to develop a program that records measurement results and automatically tells the weight differences.

Input

The input consists of multiple datasets. The first line of a dataset contains two integers N and M. N denotes the number of sample pieces ($2 \leq N \leq 100,000$). Each sample is assigned a unique number from 1 to N as an identifier. The rest of the dataset consists of M lines ($1 \leq M \leq 100,000$), each of which corresponds to either a measurement result or an inquiry. They are given in chronological order.

A measurement result has the format,

`!` a b w

which represents the sample piece numbered b is heavier than one numbered a by w micrograms ($a \neq b$). That is, $w = w_b - w_a$, where w_a and w_b are the weights of a and b, respectively. Here, w is a non-negative integer not exceeding 1,000,000.

You may assume that all measurements are exact and consistent.

An inquiry has the format,

`?` a b

which asks the weight difference between the sample pieces numbered a and b ($a \neq b$).

The last dataset is followed by a line consisting of two zeros separated by a space.

Output

For each inquiry, `?` a b, print the weight difference in micrograms between the sample pieces numbered a and b, $w_b - w_a$, followed by a newline if the weight difference can be computed based on the measurement results prior to the inquiry. The difference can be zero, or negative as well as positive. You can assume that its absolute value is at most 1,000,000. If the difference cannot be computed based on the measurement results prior to the inquiry, print `UNKNOWN` followed by a newline.

Sample Input

```
2 2
! 1 2 1
? 1 2
2 2
! 1 2 1
? 2 1
4 7
! 1 2 100
? 2 3
! 2 3 100
? 2 3
? 1 3
! 4 3 150
? 4 1
0 0
```

Output for the Sample Input

```
1
-1
UNKNOWN
100
200
-50
```

Problem G
Let There Be Light

Input: Standard Input
Time Limit: 30 seconds

Suppose that there are some light sources and many spherical balloons. All light sources have sizes small enough to be modeled as point light sources, and they emit light in all directions. The surfaces of the balloons absorb light and do not reflect light. Surprisingly in this world, balloons may overlap.

You want the total illumination intensity at an objective point as high as possible. For this purpose, some of the balloons obstructing lights can be removed. Because of the removal costs, however, there is a certain limit on the number of balloons to be removed. Thus, you would like to remove an appropriate set of balloons so as to maximize the illumination intensity at the objective point.

The following figure illustrates the configuration specified in the first dataset of the sample input given below. The figure shows the xy-plane, which is enough because, in this dataset, the z-coordinates of all the light sources, balloon centers, and the objective point are zero. In the figure, light sources are shown as stars and balloons as circles. The objective point is at the origin, and you may remove up to 4 balloons. In this case, the dashed circles in the figure correspond to the balloons to be removed.

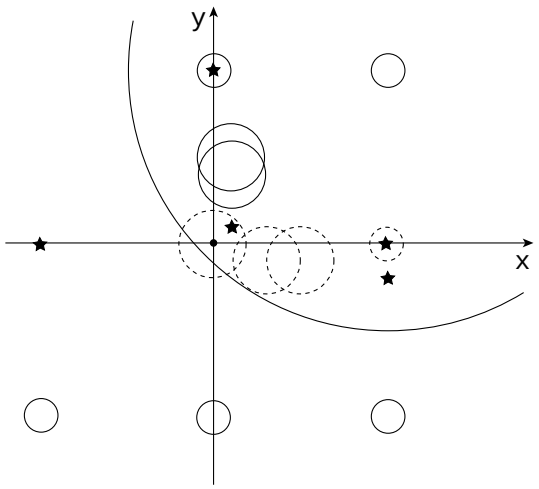

Figure G.1: First dataset of the sample input.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

$N \;\; M \;\; R$
$S_{1x} \;\; S_{1y} \;\; S_{1z} \;\; S_{1r}$
...
$S_{Nx} \;\; S_{Ny} \;\; S_{Nz} \;\; S_{Nr}$
$T_{1x} \;\; T_{1y} \;\; T_{1z} \;\; T_{1b}$
...
$T_{Mx} \;\; T_{My} \;\; T_{Mz} \;\; T_{Mb}$
$E_x \;\; E_y \;\; E_z$

The first line of a dataset contains three positive integers, N, M and R, separated by a single space. N means the number of balloons that does not exceed 2000. M means the number of light sources that does not exceed 15. R means the number of balloons that may be removed, which does not exceed N.

Each of the N lines following the first line contains four integers separated by a single space. (S_{ix}, S_{iy}, S_{iz}) means the center position of the i-th balloon and S_{ir} means its radius.

Each of the following M lines contains four integers separated by a single space. (T_{jx}, T_{jy}, T_{jz}) means the position of the j-th light source and T_{jb} means its brightness.

The last line of a dataset contains three integers separated by a single space. (E_x, E_y, E_z) means the position of the objective point.

S_{ix}, S_{iy}, S_{iz}, T_{jx}, T_{jy}, T_{jz}, E_x, E_y and E_z are greater than -500, and less than 500. S_{ir} is greater than 0, and less than 500. T_{jb} is greater than 0, and less than 80000.

At the objective point, the intensity of the light from the j-th light source is in inverse proportion to the square of the distance, namely

$$\frac{T_{jb}}{(T_{jx} - E_x)^2 + (T_{jy} - E_y)^2 + (T_{jz} - E_z)^2} \quad ,$$

if there is no balloon interrupting the light. The total illumination intensity is the sum of the above.

You may assume the following.

- The distance between the objective point and any light source is not less than 1.
- For every i and j, even if S_{ir} changes by ϵ ($|\epsilon| < 0.01$), whether the i-th balloon hides the j-th light or not does not change.

The end of the input is indicated by a line of three zeros.

Output

For each dataset, output a line containing a decimal fraction which means the highest possible illumination intensity at the objective point after removing R balloons. The output should not contain an error greater than 0.0001.

Sample Input

```
12 5 4
0 10 0 1
1 5 0 2
1 4 0 2
0 0 0 2
10 0 0 1
3 -1 0 2
5 -1 0 2
10 10 0 15
0 -10 0 1
10 -10 0 1
-10 -10 0 1
10 10 0 1
0 10 0 240
10 0 0 200
10 -2 0 52
-10 0 0 100
1 1 0 2
0 0 0
12 5 4
0 10 0 1
1 5 0 2
1 4 0 2
0 0 0 2
10 0 0 1
3 -1 0 2
5 -1 0 2
10 10 0 15
0 -10 0 1
10 -10 0 1
-10 -10 0 1
10 10 0 1
0 10 0 260
10 0 0 200
10 -2 0 52
-10 0 0 100
1 1 0 2
0 0 0
```

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

Output for the Sample Input

```
3.5
3.6
1.1666666666666667
0.0
```

Problem H
Company Organization

Input: Standard Input
Time Limit: 30 seconds

You started a company a few years ago and fortunately it has been highly successful. As the growth of the company, you noticed that you need to manage employees in a more organized way, and decided to form several groups and assign employees to them.

Now, you are planning to form n groups, each of which corresponds to a project in the company. Sometimes you have constraints on members in groups. For example, a group must be a subset of another group because the former group will consist of senior members of the latter group, the members in two groups must be the same because current activities of the two projects are closely related, the members in two groups must not be exactly the same to avoid corruption, two groups cannot have a common employee because of a security reason, and two groups must have a common employee to facilitate collaboration.

In summary, by letting X_i $(i = 1, \ldots, n)$ be the set of employees assigned to the i-th group, we have five types of constraints as follows.

1. $X_i \subseteq X_j$
2. $X_i = X_j$
3. $X_i \neq X_j$
4. $X_i \cap X_j = \emptyset$
5. $X_i \cap X_j \neq \emptyset$

Since you have listed up constraints without considering consistency, it might be the case that you cannot satisfy all the constraints. Constraints are thus ordered according to their priorities, and you now want to know how many constraints of the highest priority can be satisfied.

You do not have to take ability of employees into consideration. That is, you can assign anyone to any group. Also, you can form groups with no employee. Furthermore, you can hire or fire as many employees as you want if you can satisfy more constraints by doing so.

For example, suppose that we have the following five constraints on three groups in the order of their priorities, corresponding to the first dataset in the sample input.

- $X_2 \subseteq X_1$

- $X_3 \subseteq X_2$
- $X_1 \subseteq X_3$
- $X_1 \neq X_3$
- $X_3 \subseteq X_1$

By assigning the same set of employees to X_1, X_2, and X_3, we can satisfy the first three constraints. However, no matter how we assign employees to X_1, X_2, and X_3, we cannot satisfy the first four highest priority constraints at the same time. Though we can satisfy the first three constraints and the fifth constraint at the same time, the answer should be three.

Input

The input consists of several datasets. The first line of a dataset consists of two integers n ($2 \leq n \leq 100$) and m ($1 \leq m \leq 10000$), which indicate the number of groups and the number of constraints, respectively. Then, description of m constraints follows. The description of each constraint consists of three integers s ($1 \leq s \leq 5$), i ($1 \leq i \leq n$), and j ($1 \leq j \leq n,\ j \neq i$), meaning a constraint of the s-th type imposed on the i-th group and the j-th group. The type number of a constraint is as listed above. The constraints are given in the descending order of priority.

The input ends with a line containing two zeros.

Output

For each dataset, output the number of constraints of the highest priority satisfiable at the same time.

Sample Input

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

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

Output for the Sample Input

```
3
4
4
2
```

Problem I
Beautiful Spacing

Input: Standard Input
Time Limit: 15 seconds

Text is a sequence of words, and a word consists of characters. Your task is to put words into a grid with W columns and sufficiently many lines. For the beauty of the layout, the following conditions have to be satisfied.

1. The words in the text must be placed keeping their original order. The following figures show correct and incorrect layout examples for a 4 word text "This is a pen" into 11 columns.

T	**h**	**i**	**s**		**i**	**s**				**a**
p	**e**	**n**								

Figure I.1: A good layout.

T	**h**	**i**	**s**		**i**	**s**		**p**	**e**	**n**
a										

Figure I.2: BAD — Do not reorder words.

2. Between two words adjacent in the same line, you must place at least one space character. You sometimes have to put more than one space in order to meet other conditions.

T	**h**	**i**	**s**	**i**	**s**	**a**		**p**	**e**	**n**

Figure I.3: BAD — Words must be separated by spaces.

3. A word must occupy the same number of consecutive columns as the number of characters in it. You cannot break a single word into two or more by breaking it into lines or by inserting spaces.

T	**h**	**i**	**s**		**i**	**s**		**a**		**p**
e	**n**									

T		**h**	**i**		**s**		**i**	**s**		**a**
p	**e**	**n**								

Figure I.4: BAD — Characters in a single word must be contiguous.

4. The text must be justified to the both sides. That is, the first word of a line must start from the first column of the line, and except the last line, the last word of a line must end at the last column.

		T	**h**	**i**	**s**		**i**	**s**		**a**
p	**e**	**n**								

T	**h**	**i**	**s**		**i**	**s**		**a**		
p	**e**	**n**								

Figure I.5: BAD — Lines must be justified to both the left and the right sides.

The text is the most beautifully laid out when there is no unnecessarily long spaces. For instance, the layout in Figure I.6 has at most 2 contiguous spaces, which is more beautiful than that in Figure I.1, having 3 contiguous spaces. Given an input text and the number of columns, please find a layout such that the length of the longest contiguous spaces between words is minimum.

T	**h**	**i**	**s**			**i**	**s**			**a**
p	**e**	**n**								

Figure I.6: A good and the most beautiful layout.

Input

The input consists of multiple datasets, each in the following format.

W N
x_1 x_2 ... x_N

W, N, and x_i are all integers. W is the number of columns ($3 \leq W \leq 80,000$). N is the number of words ($2 \leq N \leq 50,000$). x_i is the number of characters in the i-th word ($1 \leq x_i \leq \frac{W-1}{2}$). Note that the upper bound on x_i assures that there always exists a layout satisfying the conditions.

The last dataset is followed by a line containing two zeros.

Output

For each dataset, print the smallest possible number of the longest contiguous spaces between words.

Sample Input

```
11 4
4 2 1 3
5 7
1 1 1 2 2 1 2
11 7
3 1 3 1 3 3 4
```

```
100 3
30 30 39
30 3
2 5 3
0 0
```

Output for the Sample Input

```
2
1
2
40
1
```

Problem J
Cubic Colonies

Input: Standard Input
Time Limit: 90 seconds

In AD 3456, the earth is too small for hundreds of billions of people to live in peace. Interstellar Colonization Project with Cubes (ICPC) is a project that tries to move people on the earth to space colonies to ameliorate the problem. ICPC obtained funding from governments and manufactured space colonies very quickly and at low cost using prefabricated cubic blocks.

The largest colony looks like a Rubik's cube. It consists of $3 \times 3 \times 3$ cubic blocks (Figure J.1A). Smaller colonies miss some of the blocks in the largest colony.

When we manufacture a colony with multiple cubic blocks, we begin with a single block. Then we iteratively glue a next block to existing blocks in a way that faces of them match exactly. Every pair of touched faces is glued.

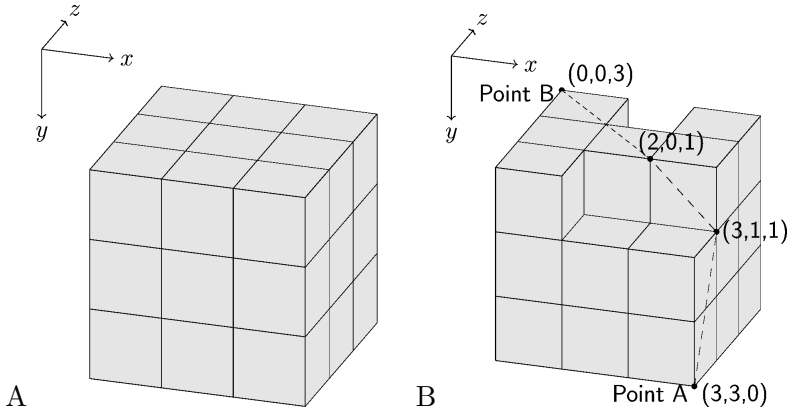

Figure J.1: Example of the largest colony and a smaller colony

However, just before the first launch, we found a design flaw with the colonies. We need to add a cable to connect two points on the surface of each colony, but we cannot change the inside of the prefabricated blocks in a short time. Therefore we decided to attach a cable on the surface of each colony. If a part of the cable is not on the surface, it would be sheared off during the launch, so we have to put the whole cable on the surface. We would like to minimize the lengths of the cables due to budget constraints. The dashed line in Figure J.1B is such an example. Write a program that, given the shape of a colony and a pair of points on its surface, calculates the length of the shortest possible cable for that colony.

Input

The input contains a series of datasets. Each dataset describes a single colony and the pair of the points for the colony in the following format.

x_1 y_1 z_1 x_2 y_2 z_2
$b_{0,0,0}$ $b_{1,0,0}$ $b_{2,0,0}$
$b_{0,1,0}$ $b_{1,1,0}$ $b_{2,1,0}$
$b_{0,2,0}$ $b_{1,2,0}$ $b_{2,2,0}$
$b_{0,0,1}$ $b_{1,0,1}$ $b_{2,0,1}$
$b_{0,1,1}$ $b_{1,1,1}$ $b_{2,1,1}$
$b_{0,2,1}$ $b_{1,2,1}$ $b_{2,2,1}$
$b_{0,0,2}$ $b_{1,0,2}$ $b_{2,0,2}$
$b_{0,1,2}$ $b_{1,1,2}$ $b_{2,1,2}$
$b_{0,2,2}$ $b_{1,2,2}$ $b_{2,2,2}$

(x_1, y_1, z_1) and (x_2, y_2, z_2) are the two distinct points on the surface of the colony, where $x_1, x_2, y_1, y_2, z_1, z_2$ are integers that satisfy $0 \leq x_1, x_2, y_1, y_2, z_1, z_2 \leq 3$. $b_{i,j,k}$ is '#' when there is a cubic block whose two diagonal vertices are (i, j, k) and $(i + 1, j + 1, k + 1)$, and $b_{i,j,k}$ is '.' if there is no block. Figure J.1A corresponds to the first dataset in the sample input, whereas Figure J.1B corresponds to the second. A cable can pass through a zero-width gap between two blocks if they are touching only on their vertices or edges. In Figure J.2A, which is the third dataset in the sample input, the shortest cable goes from the point A $(0, 0, 2)$ to the point B $(2, 2, 2)$, passing through $(1, 1, 2)$, which is shared by six blocks. Similarly, in Figure J.2B (the fourth dataset in the sample input), the shortest cable goes through the gap between two blocks not glued directly. When two blocks share only a single vertex, you can put a cable through the vertex (Figure J.2C; the fifth dataset in the sample input).

You can assume that there is no colony consisting of all $3 \times 3 \times 3$ cubes but the center cube.

Six zeros terminate the input.

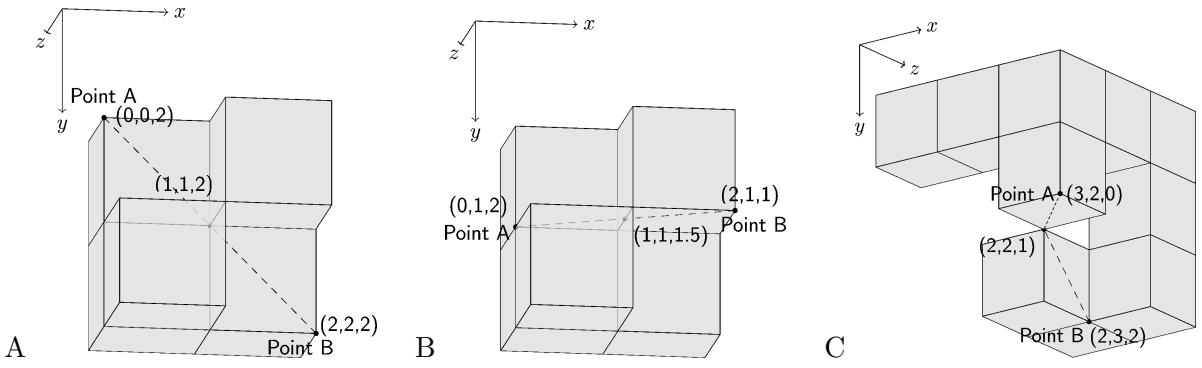

Figure J.2: Dashed lines are the shortest cables. Some blocks are shown partially transparent for illustration.

Output

For each dataset, output a line containing the length of the shortest cable that connects the two given points. We accept errors less than 0.0001. You can assume that given two points can be connected by a cable.

Sample Input

```
0 0 0 3 3 3
###
###
###
###
###
###
###
###
###
3 3 0 0 0 3
#..
###
###
###
###
###
#.#
###
###
0 0 2 2 2 2
...
...
...
.#.
#..
...
##.
##.
...
0 1 2 2 1 1
...
...
...
.#.
#..
...
##.
```

```
##.
...
3 2 0 2 3 2
###
..#
...
..#
...
.#.
..#
..#
.##
0 0 0 0 0 0
```

Output for the Sample Input

```
6.70820393249936941515
6.47870866461907457534
2.82842712474619029095
2.23606797749978980505
2.82842712474619029095
```