Problem A
Equal Sum Sets

Input: Standard Input
Time Limit: 10 seconds

Let us consider sets of positive integers less than or equal to n. Note that all elements of a set are different. Also note that the order of elements doesn't matter, that is, both $\{3, 5, 9\}$ and $\{5, 9, 3\}$ mean the same set.

Specifying the number of set elements and their sum to be k and s, respectively, sets satisfying the conditions are limited. When $n = 9$, $k = 3$ and $s = 23$, $\{6, 8, 9\}$ is the only such set. There may be more than one such set, in general, however. When $n = 9$, $k = 3$ and $s = 22$, both $\{5, 8, 9\}$ and $\{6, 7, 9\}$ are possible.

You have to write a program that calculates the number of the sets that satisfy the given conditions.

Input

The input consists of multiple datasets. The number of datasets does not exceed 100.

Each of the datasets has three integers n, k and s in one line, separated by a space. You may assume $1 \leq n \leq 20$, $1 \leq k \leq 10$ and $1 \leq s \leq 155$.

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

Output

The output for each dataset should be a line containing a single integer that gives the number of the sets that satisfy the conditions. No other characters should appear in the output.

You can assume that the number of sets does not exceed $2^{31} - 1$.

Sample Input

```
9 3 23
9 3 22
10 3 28
16 10 107
20 8 102
20 10 105
20 10 155
```

```
3 4 3
4 2 11
0 0 0
```

Output for the Sample Input

```
1
2
0
20
1542
5448
1
0
0
```

Problem B
The Last Ant

Input: Standard Input
Time Limit: 10 seconds

A straight tunnel without branches is crowded with busy ants coming and going. Some ants walk left to right and others right to left. All ants walk at a constant speed of 1 cm/s. When two ants meet, they try to pass each other. However, some sections of the tunnel are narrow and two ants cannot pass each other. When two ants meet at a narrow section, they turn around and start walking in the opposite directions. When an ant reaches either end of the tunnel, it leaves the tunnel.

The tunnel has an integer length in centimeters. Every narrow section of the tunnel is integer centimeters distant from the both ends. Except for these sections, the tunnel is wide enough for ants to pass each other. All ants start walking at distinct narrow sections. No ants will newly enter the tunnel. Consequently, all the ants in the tunnel will eventually leave it. Your task is to write a program that tells which is the last ant to leave the tunnel and when it will.

Figure B.1 shows the movements of the ants during the first two seconds in a tunnel 6 centimeters long. Initially, three ants, numbered 1, 2, and 3, start walking at narrow sections, 1, 2, and 5 centimeters distant from the left end, respectively. After 0.5 seconds, the ants 1 and 2 meet at a wide section, and they pass each other. Two seconds after the start, the ants 1 and 3 meet at a narrow section, and they turn around.

Figure B.1 corresponds to the first dataset of the sample input.

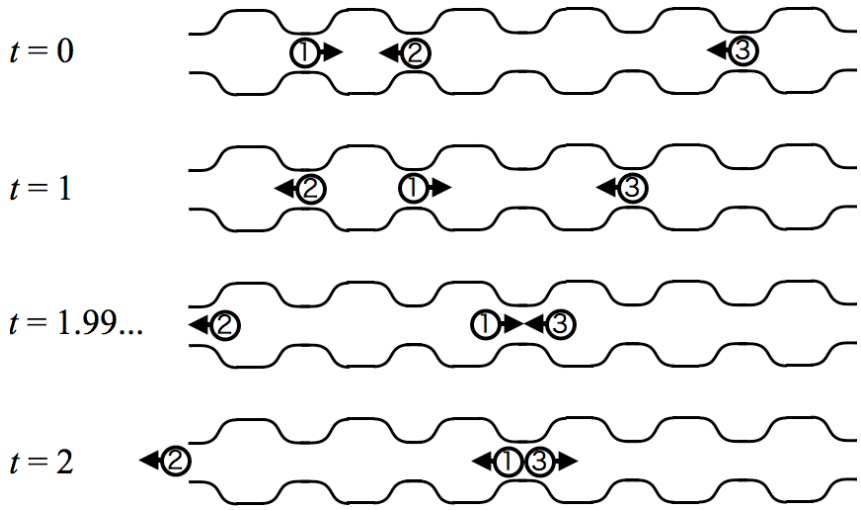


Figure B.1. Movements of ants

Input

The input consists of one or more datasets. Each dataset is formatted as follows.

$$
\begin{array}{ll}
n & l \\
d_1 & p_1 \\
d_2 & p_2 \\
\vdots & \\
d_n & p_n
\end{array}
$$

The first line of a dataset contains two integers separated by a space. n ($1 \leq n \leq 20$) represents the number of ants, and l ($n + 1 \leq l \leq 100$) represents the length of the tunnel in centimeters. The following n lines describe the initial states of ants. Each of the lines has two items, d_i and p_i, separated by a space. Ants are given numbers 1 through n. The ant numbered i has the initial direction d_i and the initial position p_i. The initial direction d_i ($1 \leq i \leq n$) is `L` (to the left) or `R` (to the right). The initial position p_i ($1 \leq i \leq n$) is an integer specifying the distance from the left end of the tunnel in centimeters. Ants are listed in the left to right order, that is, $1 \leq p_1 < p_2 < \cdots < p_n \leq l - 1$.

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

Output

For each dataset, output how many seconds it will take before all the ants leave the tunnel, and which of the ants will be the last. The last ant is identified by its number. If two ants will leave at the same time, output the number indicating the ant that will leave through the left end of the tunnel.

Sample Input

```
3 6
R 1
L 2
L 5
1 10
R 1
2 10
R 5
L 7
2 10
R 3
L 8
2 99
R 1
L 98
4 10
L 1
R 2
L 8
R 9
6 10
R 2
R 3
L 4
R 6
L 7
L 8
0 0
```

Output for the Sample Input

```
5 1
9 1
7 1
8 2
98 2
8 2
8 3
```

Problem C
Count the Regions

Input: Standard Input
Time Limit: 10 seconds

There are a number of rectangles on the x-y plane. The four sides of the rectangles are parallel to either the x-axis or the y-axis, and all of the rectangles reside within a range specified later. There are no other constraints on the coordinates of the rectangles.

The plane is partitioned into regions surrounded by the sides of one or more rectangles. In an example shown in Figure C.1, three rectangles overlap one another, and the plane is partitioned into eight regions.

Rectangles may overlap in more complex ways. For example, two rectangles may have overlapped sides, they may share their corner points, and/or they may be nested. Figure C.2 illustrates such cases.

Your job is to write a program that counts the number of the regions on the plane partitioned by the rectangles.

Input

The input consists of multiple datasets. Each dataset is formatted as follows.

$$
\begin{array}{l}
n \\
l_1 \;\; t_1 \;\; r_1 \;\; b_1 \\
l_2 \;\; t_2 \;\; r_2 \;\; b_2 \\
\vdots \\
l_n \;\; t_n \;\; r_n \;\; b_n
\end{array}
$$

A dataset starts with n ($1 \leq n \leq 50$), the number of rectangles on the plane. Each of the following n lines describes a rectangle. The i-th line contains four integers, l_i, t_i, r_i, and b_i, which are the coordinates of the i-th rectangle; (l_i, t_i) gives the x-y coordinates of the top left corner, and (r_i, b_i) gives that of the bottom right corner of the rectangle ($0 \leq l_i < r_i \leq 10^6$, $0 \leq b_i < t_i \leq 10^6$, for $1 \leq i \leq n$). The four integers are separated by a space.

The input is terminated by a single zero.

Output

For each dataset, output a line containing the number of regions on the plane partitioned by the sides of the rectangles.

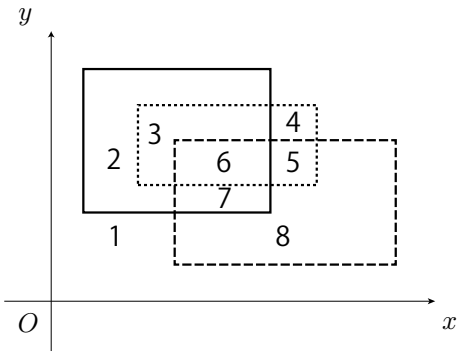


Figure C.1. Three rectangles partition the plane into eight regions. This corresponds to the first dataset of the sample input. The x- and y-axes are shown for illustration purposes only, and therefore they do not partition the plane.

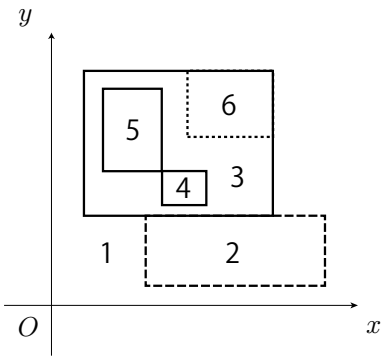


Figure C.2. Rectangles overlapping in more complex ways. This corresponds to the second dataset.

Sample Input

```
3
4 28 27 11
15 20 42 5
11 24 33 14
5
4 28 27 11
12 11 34 2
7 26 14 16
14 16 19 12
17 28 27 21
2
300000 1000000 600000 0
0 600000 1000000 300000
0
```

Output for the Sample Input

```
8
6
6
```

Problem D
Clock Hands

Input: Standard Input
Time Limit: 10 seconds

We have an analog clock whose three hands (the *second* hand, the *minute* hand and the *hour* hand) rotate quite smoothly. You can measure two angles between the second hand and two other hands.

Write a program to find the time at which "No two hands overlap each other" and "Two angles between the second hand and two other hands are equal" for the first time on or after a given time.

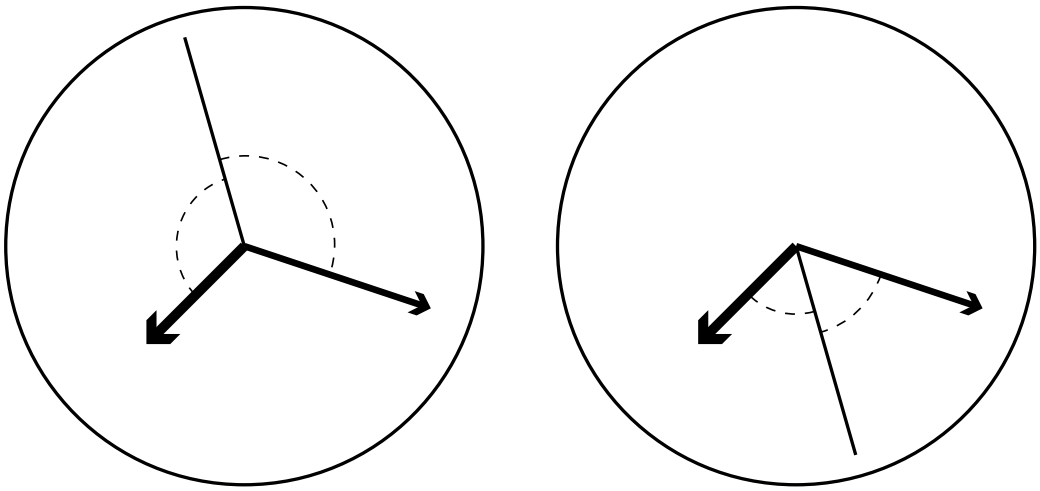


Figure D.1. Angles between the second hand and two other hands

Clocks are not limited to 12-hour clocks. The hour hand of an H-hour clock goes around once in H hours. The minute hand still goes around once every hour, and the second hand goes around once every minute. At 0:0:0 (midnight), all the hands are at the upright position.

Input

The input consists of multiple datasets. Each of the dataset has four integers H, h, m and s in one line, separated by a space. H means that the clock is an H-hour clock. h, m and s mean hour, minute and second of the specified time, respectively.

You may assume $2 \leq H \leq 100$, $0 \leq h < H$, $0 \leq m < 60$, and $0 \leq s < 60$.

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

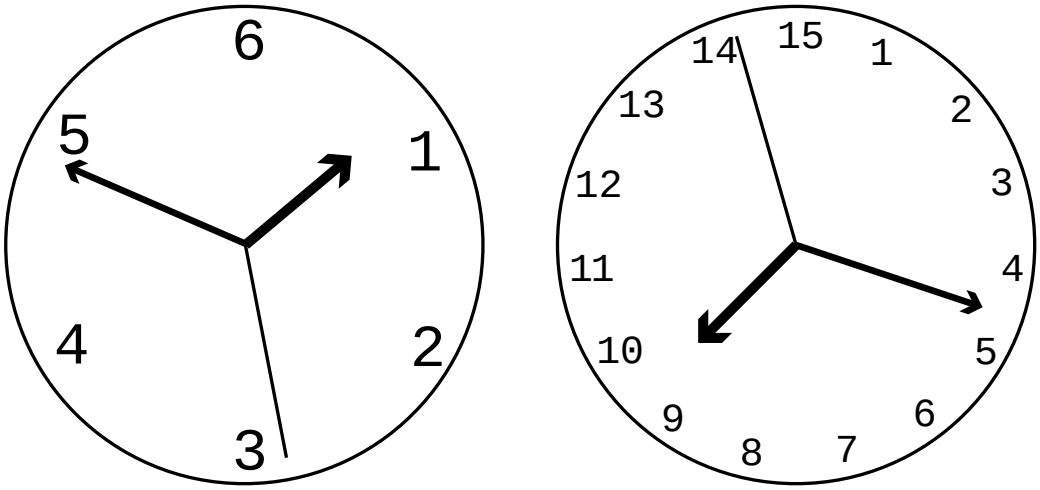


Figure D.2. Examples of H-hour clock (6-hour clock and 15-hour clock)

Output

Output the time T at which "No two hands overlap each other" and "Two angles between the second hand and two other hands are equal" for the first time on and after the specified time.

For T being h_o:m_o:s_o (s_o seconds past m_o minutes past h_o o'clock), output four non-negative integers h_o, m_o, n, and d in one line, separated by a space, where n/d is the irreducible fraction representing s_o. For integer s_o including 0, let d be 1.

The time should be expressed in the remainder of H hours. In other words, one second after $(H-1)$:59:59 is 0:0:0, not H:0:0.

Sample Input

```
12 0 0 0
12 11 59 59
12 1 56 0
12 1 56 3
12 1 56 34
12 3 9 43
12 3 10 14
12 7 17 58
12 7 18 28
12 7 23 0
12 7 23 31
2 0 38 29
2 0 39 0
2 0 39 30
2 1 6 20
```

```
2 1 20 1
2 1 20 31
3 2 15 0
3 2 59 30
4 0 28 48
5 1 5 40
5 1 6 10
5 1 7 41
11 0 55 0
0 0 0 0
```

Output for the Sample Input

```
0 0 43200 1427
0 0 43200 1427
1 56 4080 1427
1 56 47280 1427
1 57 4860 1427
3 10 18600 1427
3 10 61800 1427
7 18 39240 1427
7 18 82440 1427
7 23 43140 1427
7 24 720 1427
0 38 4680 79
0 39 2340 79
0 40 0 1
1 6 3960 79
1 20 2400 79
1 21 60 79
2 15 0 1
0 0 2700 89
0 28 48 1
1 6 320 33
1 6 40 1
1 8 120 11
0 55 0 1
```

Problem E
Dragon's Cruller

Input: Standard Input
Time Limit: 60 seconds

Dragon's Cruller is a sliding puzzle on a torus. The torus surface is partitioned into nine squares as shown in its development in Figure E.1. Here, two squares with one of the sides on the development labeled with the same letter are adjacent to each other, actually sharing the side. Figure E.2 shows which squares are adjacent to which and in which way. Pieces numbered from 1 through 8 are placed on eight out of the nine squares, leaving the remaining one empty.

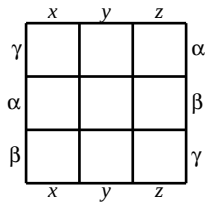
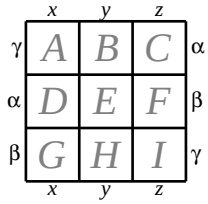


Figure E.1. A 3×3 Dragon's Cruller torus

A piece can be slid to an empty square from one of its adjacent squares. The goal of the puzzle is, by sliding pieces a number of times, to reposition the pieces from the given starting arrangement into the given goal arrangement. Figure E.3 illustrates arrangements directly reached from the arrangement shown in the center after four possible slides. The cost to slide a piece depends on the direction but is independent of the position nor the piece.

Your mission is to find the minimum cost required to reposition the pieces from the given starting arrangement into the given goal arrangement.



Square	Horizontally adjacent squares	Vertically adjacent squares
A	*B* *I*	*G* *D*
B	*C* *A*	*H* *E*
C	*D* *B*	*I* *F*
D	*E* *C*	*A* *G*
E	*F* *D*	*B* *H*
F	*G* *E*	*C* *I*
G	*H* *F*	*D* *A*
H	*I* *G*	*E* *B*
I	*A* *H*	*F* *C*

Figure E.2. Adjacency

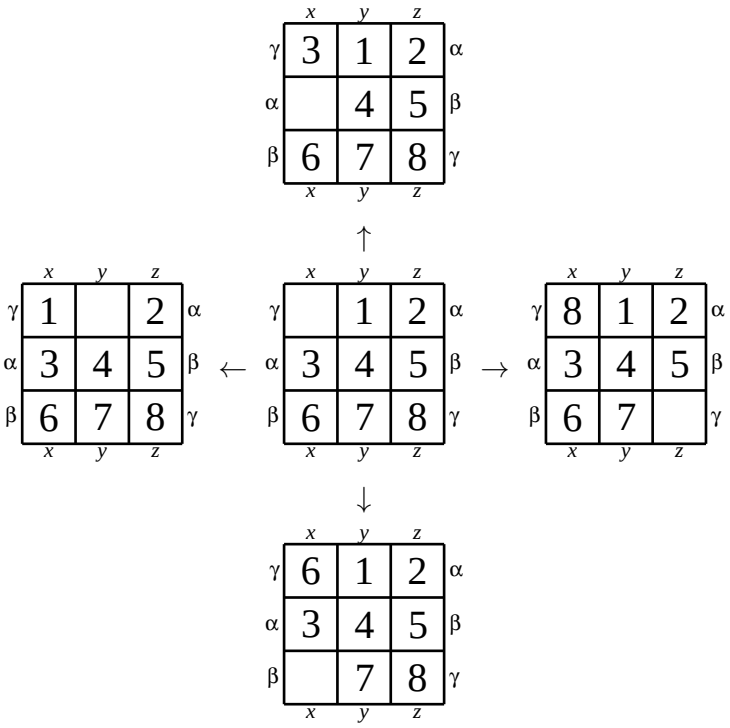


Figure E.3. Examples of sliding steps

Unlike some sliding puzzles on a flat square, it is known that any goal arrangement can be reached from any starting arrangements on this torus.

Input

The input is a sequence of at most 30 datasets.

A dataset consists of seven lines. The first line contains two positive integers c_h and c_v, which represent the respective costs to move a piece horizontally and vertically. You can assume that both c_h and c_v are less than 100. The next three lines specify the starting arrangement and the last three the goal arrangement, each in the following format.

$$\begin{array}{ccc} d_A & d_B & d_C \\ d_D & d_E & d_F \\ d_G & d_H & d_I \end{array}$$

Each line consists of three digits separated by a space. The digit d_X (X is one of A through I) indicates the state of the square X as shown in Figure E.2. Digits `1`, ..., `8` indicate that the piece of that number is on the square. The digit `0` indicates that the square is empty.

The end of the input is indicated by two zeros separated by a space.

Output

For each dataset, output the minimum total cost to achieve the goal, in a line. The total cost is the sum of the costs of moves from the starting arrangement to the goal arrangement. No other characters should appear in the output.

Sample Input

```
4 9
6 3 0
8 1 2
4 5 7
6 3 0
8 1 2
4 5 7
31 31
4 3 6
0 1 5
8 2 7
0 3 6
4 1 5
8 2 7
92 4
1 5 3
4 0 7
8 2 6
1 5 0
4 7 3
8 2 6
12 28
3 4 5
0 2 6
7 1 8
5 7 1
8 6 2
0 3 4
0 0
```

Output for the Sample Input

```
0
31
96
312
```

Problem F
Directional Resemblance

Input: Standard Input
Time Limit: 60 seconds

Vectors have their directions and two vectors make an angle between them. Given a set of three-dimensional vectors, your task is to find the pair among them that makes the smallest angle.

Input

The input is a sequence of datasets. A dataset specifies a set of three-dimensional vectors, some of which are directly given in the dataset and the others are generated by the procedure described below.

Each dataset is formatted as follows.

$$
\begin{array}{llll}
m & n & S & W \\
x_1 & y_1 & z_1 & \\
x_2 & y_2 & z_2 & \\
\vdots & & & \\
x_m & y_m & z_m &
\end{array}
$$

The first line contains four integers m, n, S, and W.

The integer m is the number of vectors whose three components are directly specified in the dataset. Starting from the second line, m lines have the three components of m vectors. The i-th line of which indicates the vector $v_i = (x_i, y_i, z_i)$. All the vector components are positive integers less than or equal to 100.

The integer n is the number of vectors generated by the following procedure.

```
int g = S;
for(int i=m+1; i<=m+n; i++) {
  x[i] = (g/7)    %100 + 1;
  y[i] = (g/700)  %100 + 1;
  z[i] = (g/70000)%100 + 1;
  if( g%2 == 0 ) { g = (g/2); }
  else           { g = (g/2) ^ W; }
}
```

For $i = m + 1, \ldots, m + n$, the i-th vector v_i of the set has three components `x[`i`]`, `y[`i`]`, and `z[`i`]` generated by this procedure.

Here, values of `S` and `W` are the parameters S and W given in the first line of the dataset. You may assume $1 \leq S \leq 10^9$ and $1 \leq W \leq 10^9$.

The total number of vectors satisfies $2 \leq m + n \leq 12 \times 10^4$. Note that exactly the same vector may be specified twice or more in a single dataset.

A line containing four zeros indicates the end of the input. The total of $m+n$ for all the datasets in the input never exceeds 16×10^5.

Output

For each dataset, output the pair of vectors among the specified set with the smallest non-zero angle in between. It is assured that there are at least two vectors having different directions.

Vectors should be indicated by their three components. The output for a pair of vectors v_a and v_b should be formatted in a line as follows.

$$x_a \quad y_a \quad z_a \quad x_b \quad y_b \quad z_b$$

Two vectors (x_a, y_a, z_a) and (x_b, y_b, z_b) are ordered in the dictionary order, that is, $v_a < v_b$ if $x_a < x_b$, or if $x_a = x_b$ and $y_a < y_b$, or if $x_a = x_b$, $y_a = y_b$ and $z_a < z_b$. When a vector pair is output, the smaller vector in this order should be output first.

If more than one pair makes the equal smallest angle, the pair that is the smallest among them in the dictionary order between vector pairs should be output. The pair (v_i, v_j) is smaller than the pair (v_k, v_l) if $v_i < v_k$, or if $v_i = v_k$ and $v_j < v_l$.

Sample Input

```
4 0 2013 1124
1 1 1
2 2 2
2 1 1
1 2 1
2 100 131832453 129800231
42 40 46
42 40 46
0 100000 7004674 484521438
0 0 0 0
```

Output for the Sample Input

```
1 1 1 1 2 1
```

```
42 40 46 83 79 91
92 92 79 99 99 85
```

Problem G
Longest Chain

Input: Standard Input
Time Limit: 60 seconds

Let us compare two triples $a = (x_a, y_a, z_a)$ and $b = (x_b, y_b, z_b)$ by a partial order $\prec$ defined as follows.

$$a \prec b \iff x_a < x_b \text{ and } y_a < y_b \text{ and } z_a < z_b$$

Your mission is to find, in the given set of triples, the longest ascending series $a_1 \prec a_2 \prec \cdots \prec a_k$.

Input

The input is a sequence of datasets, each specifying a set of triples formatted as follows.

$$\begin{array}{llll} m & n & A & B \\ x_1 & y_1 & z_1 & \\ x_2 & y_2 & z_2 & \\ \vdots & & & \\ x_m & y_m & z_m & \end{array}$$

Here, m, n, A and B in the first line, and all of x_i, y_i and z_i $(i = 1, \ldots, m)$ in the following lines are non-negative integers.

Each dataset specifies a set of $m + n$ triples. The triples p_1 through p_m are explicitly specified in the dataset, the i-th triple p_i being (x_i, y_i, z_i). The remaining n triples are specified by parameters A and B given to the following generator routine.

```
int a = A, b = B, C = ~(1<<31), M = (1<<16)-1;
int r() {
  a = 36969 * (a & M) + (a >> 16);
  b = 18000 * (b & M) + (b >> 16);
  return (C & ((a << 16) + b)) % 1000000;
}
```

Repeated $3n$ calls of `r()` defined as above yield values of x_{m+1}, y_{m+1}, z_{m+1}, x_{m+2}, y_{m+2}, z_{m+2}, ..., x_{m+n}, y_{m+n}, and z_{m+n}, in this order.

You can assume that $1 \le m + n \le 3 \times 10^5$, $1 \le A, B \le 2^{16}$, and $0 \le x_k, y_k, z_k < 10^6$ for $1 \le k \le m + n$.

The input ends with a line containing four zeros. The total of $m + n$ for all the datasets does not exceed 2×10^6.

Output

For each dataset, output the length of the longest ascending series of triples in the specified set. If $p_{i_1} \prec p_{i_2} \prec \cdots \prec p_{i_k}$ is the longest, the answer should be k.

Sample Input

```
6 0 1 1
0 0 0
0 2 2
1 1 1
2 0 2
2 2 0
2 2 2
5 0 1 1
0 0 0
1 1 1
2 2 2
3 3 3
4 4 4
10 0 1 1
3 0 0
2 1 0
2 0 1
1 2 0
1 1 1
1 0 2
0 3 0
0 2 1
0 1 2
0 0 3
0 10 1 1
0 0 0 0
```

Output for the Sample Input

```
3
5
1
3
```

Problem H
Don't Burst the Balloon

Input: Standard Input
Time Limit: 30 seconds

An open-top box having a square bottom is placed on the floor. You see a number of needles vertically planted on its bottom.

You want to place a largest possible spheric balloon touching the box bottom, interfering with none of the side walls nor the needles.

Java Specific: Submitted Java programs may not use "java.awt.geom.Area". You may use it for your debugging purposes.

Figure H.1 shows an example of a box with needles and the corresponding largest spheric balloon. It corresponds to the first dataset of the sample input below.

Input

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

$$
\begin{array}{ll}
n & w \\
x_1 & y_1 \quad h_1 \\
\vdots & \\
x_n & y_n \quad h_n
\end{array}
$$

The first line of a dataset contains two positive integers, n and w, separated by a space. n represents the number of needles, and w represents the height of the side walls.

The bottom of the box is a 100×100 square. The corners of the bottom face are placed at positions (0, 0, 0), (0, 100, 0), (100, 100, 0), and (100, 0, 0).

Each of the n lines following the first line contains three integers, x_i, y_i, and h_i. $(x_i, y_i, 0)$ and h_i represent the base position and the height of the i-th needle. No two needles stand at the same position.

You can assume that $1 \leq n \leq 10$, $10 \leq w \leq 200$, $0 < x_i < 100$, $0 < y_i < 100$ and $1 \leq h_i \leq 200$. You can ignore the thicknesses of the needles and the walls.

The end of the input is indicated by a line of two zeros. The number of datasets does not exceed 1000.

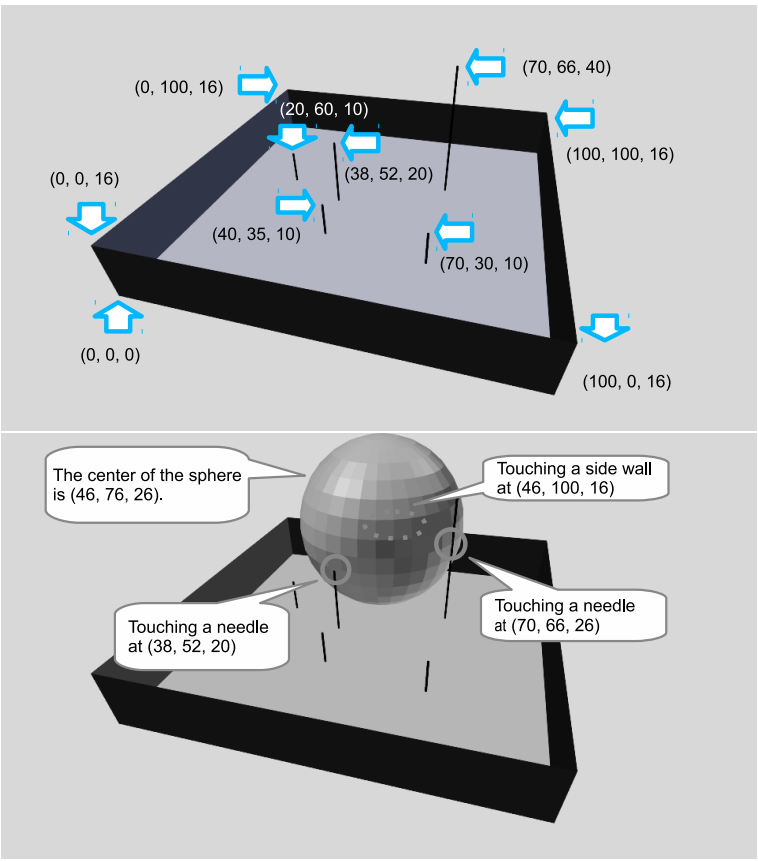


Figure H.1. The upper shows an example layout and the lower shows the largest spheric balloon that can be placed.

Output

For each dataset, output a single line containing the maximum radius of a balloon that can touch the bottom of the box without interfering with the side walls or the needles. The output should not contain an error greater than 0.0001.

Sample Input

```
5 16
70 66 40
38 52 20
40 35 10
70 30 10
20 60 10
1 100
54 75 200
1 10
90 10 1
1 11
54 75 200
3 10
53 60 1
61 38 1
45 48 1
4 10
20 20 10
20 80 10
80 20 10
80 80 10
0 0
```

Output for the Sample Input

```
26.00000
39.00000
130.00000
49.49777
85.00000
95.00000
```

Problem I
Hidden Tree

Input: Standard Input
Time Limit: 20 seconds

Consider a binary tree whose leaves are assigned integer weights. Such a tree is called *balanced* if, for every non-leaf node, the sum of the weights in its left subtree is equal to that in the right subtree. For instance, the tree in the following figure is balanced.

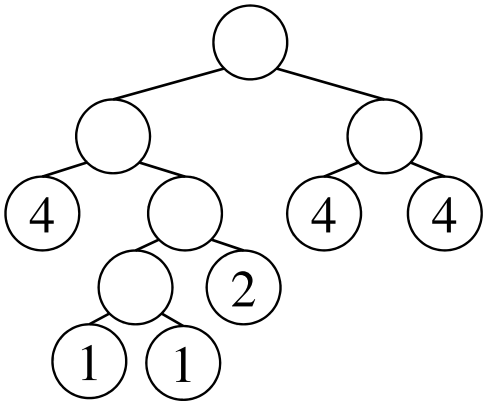


Figure I.1. A balanced tree

A balanced tree is said to be *hidden* in a sequence A, if the integers obtained by listing the weights of all leaves of the tree from left to right form a subsequence of A. Here, a subsequence is a sequence that can be derived by deleting zero or more elements from the original sequence without changing the order of the remaining elements.

For instance, the balanced tree in the figure above is hidden in the sequence 3 4 1 3 1 2 4 4 6, because 4 1 1 2 4 4 is a subsequence of it.

Now, your task is to find, in a given sequence of integers, the balanced tree with the largest number of leaves hidden in it. In fact, the tree shown in Figure I.1 has the largest number of leaves among the balanced trees hidden in the sequence mentioned above.

Input

The input consists of multiple datasets. Each dataset represents a sequence A of integers in the format

N
$A_1\ A_2\ \ldots\ A_N$

where $1 \leq N \leq 1000$ and $1 \leq A_i \leq 500$ for $1 \leq i \leq N$. N is the length of the input sequence, and A_i is the i-th element of the sequence.

The input ends with a line consisting of a single zero. The number of datasets does not exceed 50.

Output

For each dataset, find the balanced tree with the largest number of leaves among those hidden in A, and output, in a line, the number of its leaves.

Sample Input

```
9
3 4 1 3 1 2 4 4 6
4
3 12 6 3
10
10 9 8 7 6 5 4 3 2 1
11
10 9 8 7 6 5 4 3 2 1 1
8
1 1 1 1 1 1 1 1
0
```

Output for the Sample Input

```
6
2
1
5
8
```

Problem J

C(O|W|A*RD*|S)* CROSSWORD Puzzle

Input: Standard Input
Time Limit: 90 seconds

The first crossword puzzle was published on December 21, 1913 by Arthur Wynne. To celebrate the centennial of his great-great-grandfather's invention, John "Coward" Wynne[1] was struggling to make crossword puzzles. He was such a coward that whenever he thought of a tricky clue for a word, he couldn't stop worrying if people would blame him for choosing a bad clue that could never mean that word. At the end of the day, he cowardly chose boring clues, which made his puzzles less interesting.

One day, he came up with a brilliant idea: puzzles in which word meanings do not matter, and yet interesting. He told his idea to his colleagues, who admitted that the idea was intriguing. They even named his puzzles "Coward's Crossword Puzzles" after his nickname.

However, making a Coward's crossword puzzle was not easy. Even though he did not have to think about word meanings, it was not easy to check if a puzzle has one and only one set of answers. As the day of the centennial anniversary was approaching, John started worrying if he would not be able to make interesting ones in time. Let's help John by writing a program that solves Coward's crossword puzzles.

Each puzzle consists of $h \times w$ cells along with h across clues and w down clues. The clues are regular expressions written in a pattern language whose BNF syntax is given as in the table below.

```
clue       ::= "^" pattern "$"
pattern    ::= simple | pattern "|" simple
simple     ::= basic | simple basic
basic      ::= elementary | elementary "*"
elementary ::= "." | "A" | "B" | ... | "Z" | "(" pattern ")"
```

Table J.1. BNF syntax of the pattern language.

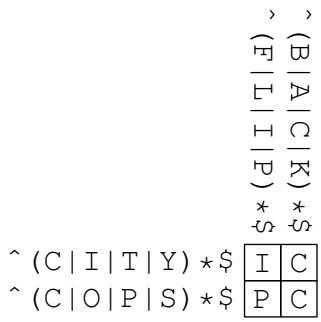
The clues (as denoted by p and q below) match words (as denoted by s below) according to the following rules.

- `^`p`$` matches s if p matches s.
- p`|`q matches a string s if p and/or q matches s.

[1] All characters appearing in this problem, except for Arthur Wynne, are fictitious. Any resemblance to real persons, living or dead, is purely coincidental.

- pq matches a string s if there exist s_1 and s_2 such that $s_1 s_2 = s$, p matches s_1, and q matches s_2.
- $p*$ matches a string s if s is empty, or there exist s_1 and s_2 such that $s_1 s_2 = s$, p matches s_1, and $p*$ matches s_2.
- Each of `A`, `B`, ..., `Z` matches the respective letter itself.
- (p) matches s if p matches s.
- `.` is the shorthand of `(A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z)`.

Below is an example of a Coward's crossword puzzle with the answers filled in the cells.



Java Specific: Submitted Java programs may not use classes in the java.util.regex package.

C++ Specific: Submitted C++ programs may not use the std::regex class.

Input

The input consists of multiple datasets, each of which represents a puzzle given in the following format.

h w
p_1
p_2
⋮
p_h
q_1
q_2
⋮
q_w

Here, h and w represent the vertical and horizontal numbers of cells, respectively, where $2 \leq h, w \leq 4$. The p_i and q_j are the across and down clues for the i-th row and j-th column, respectively. Any clue has no more than 512 characters.

The last dataset is followed by a line of two zeros. You may assume that there are no more than 30 datasets in the input.

Output

For each dataset, when the puzzle has a unique set of answers, output it as h lines of w characters. When the puzzle has no set of answers, or more than one set of answers, output "`none`" or "`ambiguous`" without double quotations, respectively.

Sample Input

```
2 2
^(C|I|T|Y)*$
^(C|O|P|S)*$
^(F|L|I|P)*$
^(B|A|C|K)*$
2 2
^HE|LL|O*$
^(P|L|E|A|S|E)*$
^(H|L)*$
^EP|IP|EF$
4 4
^LONG|TALL|S*ALLY$
^(R*EV*|OL*U(TIO)*N)*$
^(STRAWBERRY|F*I*E*L*D*S*|FOREVER)*$
^P.S.|I|LOVE|YOU$
^(RE|A|L)((L|OV*E)*)$
^(LUC*Y*|IN.THE.SKY)(WITH|DI*A*M*ON*D*S*)$
^(IVE*|GOT|A|F*E*E*L*I*N*G*)*$
^YEST*E*R*D*A*Y*$
2 3
^(C|P)(OL|AS)$
^(LU|TO)(X|R)$
^CT|PL$
^OU|AO$
^SR|LX$
2 2
^T*|(HI)|S*$
^SE|NT|EN|CE$
^IS$
^(F|A|L|S|E)*$
2 4
^ARKA|BARB|COLU$
^NSAS|ADOS|MBIA$
^..$
^..$
```

```
^KA|RO|LI$
^AS|BA|US$
0 0
```

Output for the Sample Input

```
IC
PC
HE
LP
ALLY
OUNE
EDIS
LOVE
ambiguous
none
ARKA
NSAS
```